Home Computer Texas Instruments TI-99/4A

Manuale d'uso

AVVERTENZA IMPORTANTE

TEXAS INSTRUMENTS NON FORNISCE GARANZIA, NE ESPRESSA NE IMPLICITA, NEI CONFRONTI DI QUESTO MANUALE E DEL SOFTWARE IN ESSO DESCRITTO, DELLA SUA QUALITA', PRESTAZIONI, COMMERCIABILITA' E RISPONDENZE ALLE NECESSITA' DI UNA PARTICOLARE APPLICAZIONE. TEXAS INSTRUMENTS FORNISCE QUESTI MATERIALI SOLAMENTE SULLA BASE DEL "COSI' COM'E"". IN NESSUN CASO TEXAS INSTRUMENTS SARA' RESPONSABILE VERSO TERZI PER DANNI DIRETTAMENTE, COLLATERALMENTE O INCIDENTALMENTE COLLEGABILI O DERIVANTI DALL'ACQUISTO O DALL'USO DEI SUDDETTI TESTI E PROGRAMMI; L'UNICA ED ESCLUSIVA RESPONSABILITA' DELLA TEXAS INSTRUMENTS, SENZA TENER CONTO DEL TIPO DI USO, NON ECCEDERA' IL VALORE DEL PREZZO DI ACQUISTO DI QUESTO "HOME COMPUTER". INOLTRE TEXAS INSTRUMENTS NON SARA' RESPONSABILE PER ALCUN RECLAMO, DI QUALSIASI TIPO ESSO SIA, DA PARTE DI CHIUNQUE NEI CONFRONTI DELL'UTILIZZATORE DI QUESTI TESTI E PROGRAMMI.

Home Computer Texas Instruments TI-99/4A

Manuale d'uso

Un manuale completo e dettagliato per utilizzare efficacemente il vostro Home Computer Texas Instruments.

Indice

I. GENERALITA'

Indice

Indice

Verso un nuovo mondo........

Aiutarvi nell'educazione dei vostri figli, nell'aggiornamento delle vostre stesse conoscenze, fare da passatempo creativo per tutta la famiglia, alleggerirvi nel vostro lavoro, in casa e in ufficio; tutto questo può fare l'Home Computer TI 99/4A e introdurvi così in un nuovo mondo.
Dimenticate tutto quello che sapevate o pensavate dei calcolatori e avanzate con fiducia verso questo nuovo mondo con il vostro Home Computer della Texas Instruments.

Questa guida vi introduce nel nuovo emozionante mondo dei calcolatori.
Il TI 99/4A vi può aiutare nell'apprendimento delle lingue o della matematica (a scuola o in ufficio) e, più in generale, nell'uso dei calcolatori.
Vi mette a disposizione programmi di giochi divertenti e istruttivi (che sviluppano capacità logico-strategiche) e programmi musicali, per passare in modo creativo il vostro tempo libero. Nella economia famigliare vi può essere di aiuto per tenere il bilancio.
Vi permette di ricavare dal vostro normale televisore un "sistema di comunicazione". Non avete bisogno di conoscere il calcolatore; vi basta seguire le facili istruzioni di questo manuale, le descrizioni dei moduli di programma e del dialogo con il video.

Sistema a Moduli

L'esclusivo Sistema a Moduli di Comando* pre-programmati in "Solid State Software™" vi farà divertire con il vostro calcolatore. Dovete solo inserirli e lanciarli. Potete scegliere in una vasta gamma offerta dalla Texas Instruments.

Programmi su nastro e floppy disk

Oltre ai moduli di programma, c'è una vasta scelta di programmi su nastro o floppy disk, anch'essi pronti per l'uso, senza nessun intervento di programmazione da parte vostra. Chiedete informazioni al vostro rivenditore.

Linguaggi di programmazione

- **TI BASIC**, che fa parte integrante del vostro Home Computer, è un semplice ma potente linguaggio di programmazione. Con esso potete sviluppare i vostri programmi, con grafica a colori, musica e persino parole.

- **TI-LOGO*** è un linguaggio di facile comprensione, che consente a studenti di qualsiasi livello di comunicare con il calcolatore. Crea un ambiente idoneo ad un apprendimento il più possibile naturale di discipline "formali" come la matematica.

- **TI EXTENDED BASIC*** è un potente linguaggio ad alto livello, che offre diverse prestazioni non disponibili in TI BASIC. Vi rimandiamo al Capitolo BASIC per una breve spiegazione.

- **UCSD PASCAL*** è un linguaggio altamente strutturato ed efficiente, più rapido, più logico e più potente del BASIC. E' concepito per lo studente di programmazione, e fornisce una notevole potenza per applicazioni di software avanzato. I programmi PASCAL sono strutturati a blocchi, in modo che le parti di un programma logicamente separate, sono sviluppate una alla volta come unità a sè stanti.

- **TMS 9900 EDITOR/ASSEMBLER***. Con l'aiuto di questo potete programmare direttamente in linguaggio macchina. E' più difficile che con gli altri linguaggi, ma i programmi sono molto veloci. Tra l'altro, la programmazione in linguaggio assembler ha una funzione didattica in se stessa ed è il modo migliore per imparare come lavora il vostro calcolatore.

* Venduti separatamente

Verso un nuovo mondo........

COME USARE QUESTO MANUALE

Il manuale è organizzato nel modo seguente
- Informazioni di carattere generale sulla tastiera e sui collegamenti
- Accessori
- Introduzione alla programmazione
- Presentazione particolareggiata del TI BASIC
- Programmi applicativi
- Terminologia e definizioni

Ed ora : Buona fortuna nella vostra avventura in questo nuovo mondo dell'apprendimento con l'Home Computer TI 99/4A della Texas Instruments.

La tastiera

Una panoramica sulla tastiera

Osserviamo da vicino la tastiera.

La tastiera è simile a quella di una macchina da scrivere standard, con tasti di diversi tipi. Premendo un tasto compare sullo schermo il carattere minuscolo, o quello inferiore; se contemporaneamente si preme il tasto **SHIFT** compare il carattere maiuscolo, o quello superiore.

Alcuni tasti servono per funzioni speciali, come vedremo ora.

RIPETIZIONE AUTOMATICA
Nel TI BASIC è prevista una funzione di ripetizione automatica; se tenete premuta la **BARRA SPAZIATRICE** o qualsiasi altro tasto per più di un secondo, il carattere viene ripetuto finchè non lo rilasciate.

IL TASTO ALPHA LOCK
Quando questo tasto è premuto, i caratteri alfabetici vengono stampati maiuscoli, mentre quelli numerici e di punteggiatura restano inalterati. Premendo una seconda volta questo tasto, la tastiera ritorna al modo normale.

Nota: per il calcolatore non potete usare il tasto "I" come numero 1 e non dovete mai sostituire la lettera "O" con uno zero.
Per distinguerli tra di loro la "O" sullo schermo compare con gli angoli acuti, lo zero con gli angoli smussati.

I TASTI DEI SIMBOLI E DELLA PUNTEGGIATURA
La tastiera ha tutti i tasti di una normale macchina da scrivere, più alcuni altri tipici delle applicazioni su calcolatore. Per battere il simbolo che si trova in basso sul tasto, premete il tasto; per quello che si trova in alto, tenete premuto **SHIFT** e premete il tasto. Notate che sul fronte di alcuni tasti compaiono punti e simboli. Per questi dovete tenere premuto il tasto **FCTN** e premerli.

TASTI DI FUNZIONE
Alcuni tasti hanno una funzione diversa a seconda che vengano usati in TI BASIC, con alcuni Moduli di Comando, o in altre applicazioni. Sono descritti dettagliatamente nel paragrafo opportuno di questo Manuale, o nei Manuali dei vari Moduli.

Per attivare qualsiasi funzione, tranne la **ENTER**, tenete premuto **FCTN** e premete il tasto con la lettera o il numero corrispondente.

Scheda/Cornice per tasti a due livelli di funzione

Il calcolatore vi viene fornito con una striscia a due livelli da sovrapporre alla tastiera, per identificare più facilmente i tasti da usare in combinazione con **FCTN** e **CTRL**.

Le funzioni in alto, identificate con il punto rosso, si chiamano tasti di controllo. Per accedervi, tenete premuto il tasto **CTRL**, segnato con un punto rosso, mentre premete l'opportuno tasto alfabetico o numerico. Le funzioni in basso, identificate dal punto grigio, si richiamano tenendo premuto il tasto **FCTN**, anch'esso segnato con un punto grigio, e premendo il tasto opportuno.

FCTN=(QUIT)

Premendo **QUIT** in qualsiasi momento il calcolatore ritorna al "Titolo Principale". *Nota*-Premendo **QUIT** si cancellano tutti i dati e i programmi che avete inserito.

ENTER

In genere, premendo **ENTER** dite al calcolatore di accettare le informazioni che avete appena battuto. Altre funzioni saranno spiegate nei manuali relativi.

FCTN← (LEFT) [a sinistra]

Premendo il tasto con la freccia a sinistra il cursore si sposta a sinistra. Il cursore non cancella nè modifica i caratteri sullo schermo.

FCTN → (RIGHT) [**a destra**]
Premendo il tasto con la freccia a destra, il cursore si sposta a destra. Come nel caso precedente, non altera in nessun modo i caratteri su cui passa.

FCTN ↑ (UP) [**in su**]
FCTN ↓ (DOWN) [**in giù**]
Questi tasti hanno diverse funzioni, a seconda della applicazione in cui sono usati. Vi rimandiamo al paragrafo TI BASIC per una spiegazione esauriente del loro utilizzo.

FCTN1 (INS) [**inserire**]
E' usato per inserire una lettera, un numero o un altro carattere nelle linee che battete.

FCTN2 (DEL) [**cancellare**]
E' usato per cancellare una lettera, un numero o un altro carattere dalla linea che battete.

FCTN 3 (ERASE) [**eliminare**]
Premendo questo tasto prima di **ENTER** viene cancellata la linea che state battendo.

FCTN 4 (CLEAR) [**annullare**]
In genere viene usato per cancellare dallo schermo tutte le informazioni che avete inserito (prima di premere ENTER). In TI BASIC ha altre funzioni, che sono descritte nel paragrafo "Tasti Speciali" del BASIC.

Ci sono poi altri tasti speciali, tra cui

FCTN 5 **(BEGIN)** [**iniziare**]
FCTN 6 **(PROC'D)** [**proseguire**]
FCTN 7 **(AID)** [**aggiungere**]
FCTN 8 **(REDO)** [**rifare**]
FCTN 9 **(BACK)** [**tornare indietro**]

TASTI DI CONTROLLO
Il calcolatore ha inoltre dei tasti di controllo che servono soprattutto per le telecomunicazioni. Per averne l'elenco guardate i"tasti di controllo" listati nel sottoprogramma KEY. Per inserire un carattere di controllo, premete il tasto opportuno insieme con il CTRL.

TASTI PER OPERAZIONI ARITMETICHE
Sono usati per fare eseguire al calcolatore somme, sottrazioni, moltiplicazioni, divisioni ed elevamenti a potenza.

I simboli per queste operazioni sono quelli soliti, tranne forse quelli della moltiplicazione e della divisione.

+ Addizione
- Sottrazione
* Moltiplicazione
/ Divisione
= Uguale

Anche il carattere (∧) è usato per operazioni matematiche.

SHIFT ∧
Serve per fare eseguire al computer un elevamento a potenza. Poichè 5^3 non può essere scritto sul vostro schermo, il calcolatore interpreta 5∧3 come 5 elevato alla terza.

I seguenti tasti sono usati per rappresentare relazioni matematiche in TI BASIC.
SHIFT> "Maggiore di", è un simbolo usato per confrontare due quantità
SHIFT< "Minore di", anche questo è usato per con frontare due quantità

BARRA SPAZIATRICE
E' la barra in basso nella tastiera, funziona come quella di una normale macchina da scrivere. Quando la premete, il calcolatore lascia uno spazio tra due parole, lettere o numeri.

Può anche essere usata per cancellare caratteri dallo schermo.(Vedi il paragrafo "Correzione degli errori").

Espansione del Sistema

Correzione degli errori

Per correggere un errore di battitura prima di battere **ENTER**, spostate il cursore fino al carattere che volete correggere, con la freccia a sinistra. Ribattete il o i caratteri corretti, e riportate il cursore alla fine della parola o della frase che stavate scrivendo, con la freccia a destra.

Potete correggere gli errori con la **BARRA SPAZIATRICE**. Tornate con la freccia a sinistra al punto in cui volete cominciare a cancellare, e premete la **BARRA SPAZIATRICE** per spostare il cursore sui caratteri dello schermo. I caratteri sono cancellati.

In certe applicazioni potete fare delle correzioni con i tasti **DEL** e **INS**.

Espansione del Sistema

E' disponibile una vasta gamma di dispositivi per fare un uso efficiente del vostro calcolatore. Questi espandono l'unità base, dandovi la possibilità di costruire il sistema secondo le vostre esigenze.
Sia che vogliate espandere la memoria, o che vogliate stampare i risultati, o infine far "parlare "il vostro calcolatore, per ognuna di queste esigenze c'è il dispositivo adatto per il TI 99/4A.
Il calcolatore cresce con le vostre esigenze.

Controlli a distanza via cavo*

I leggeri e compatti Controlli a distanza via cavo rendono più indipendenti e versatili le applicazioni di giochi, grafica e musica, eliminando la necessità della tastiera. Potete usarli con alcune applicazioni software o con i vostri stessi programmi in TI BASIC.

Sintetizzatore vocale*

Dà al vostro calcolatore una voce personale. Le applicazioni del calcolatore assumono un nuovo interesse, con l'emissione di parole, frasi e interi periodi parlati. C'è uno speciale Modulo di Comando* per attivarlo, da inserire nella console del calcolatore. Potete usare lo "Speech Editor", il "Terminal Emulator II", o qualsiasi altro modulo progettato per la voce.

Sistema di memoria a dischi*

E' una memoria di massa costituita da una Scheda di Controllo per l'Unità a Dischi TI, e da uno a tre Unità a Dischi. Con questo sistema, potete conservare il vostro programma per usarlo in un secondo tempo, oppure usare programmi disponibili su disco. Inoltre ci sono dei Moduli di Comando progettati per farvi registrare dati e risultati dei vostri calcoli.

Il Modulo di Comando "Disk Manager" (Gestione del Disco)* è fornito con ogni Controllo per Unità a Dischi e vi consente di catalogare il contenuto di un disco, di dare un nome a un disco o a un file, di cancellare file, duplicare un disco o un file, proteggere i vostri file e controllare le operazioni sui dischi.

La Stampante*

Quando al vostro calcolatore è collegata la stampante potete ottenere una copia stampata del vostro programma e dei dati, che vi può essere di aiuto per rivedere programmi particolarmente lunghi, o per tenere un archivio dei programmi e dei risultati. Inoltre la stampante può essere usata in alcune applicazioni software per stampare ciò che compare sul video o per generare stampe di elenchi e di documenti.

La stampante stampa fino a 32 carattei per linea.

Interfaccia TI RS232*

Questa interfaccia vi consente di collegare al vostro calcolatore una vasta gamma di dispositivi compatibili con EIA RS232C. Potete stampare il listing di un programma, utilizzare un modem per telecomunicazioni, stampare grafici su di un plotter, e altro ancora.

Interfaccia Telefonica (Modem)*

Consente la comunicazione su linee telefoniche tra il vostro calcolatore ed un altro analogo. Se avete un modulo di Comando TI per le telecomunicazioni potete abbonarvi a diversi Sistemi Informativi.

* in vendita separatamente

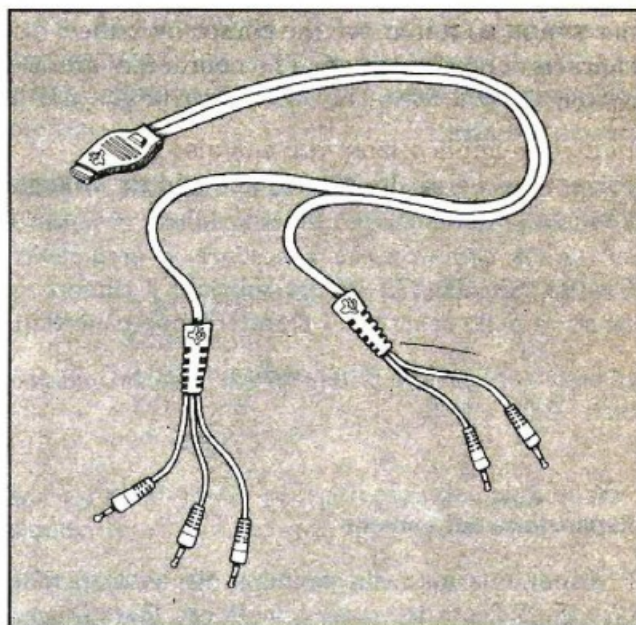
Espansione del Sistema

Unità di Espansione della Memoria*

L'Unità di Espansione della Memoria aggiunge 32K di RAM alla memoria del calcolatore. Inoltre aumenta il numero di dispositivi che possono essere collegati al calcolatore. (*Nota*-L'Unità di Espansione della Memoria richiede l'uso di un Modulo di Comando o di un dispositivo apposito. Il TI BASIC del calcolatore non può utilizzarla)

Connettore per interfaccia col Registratore a Cassette*

Potete espandere ulteriormente il vostro sistema con un registratore a cassetta. TI BASIC vi consente di registrare e recuperare i dati che avete inserito nel calcolatore (programmi, dati numerici, etc.). Registrando i dati su di un registratore, potete *salvarli* in modo permanente. In un secondo tempo potete *caricare* i dati dal registratore in memoria, se volete usarli ancora. Molti Moduli di Comando salvano e poi recuperano i dati usati nel modulo stesso.

Potete usare uno o due registratori, due sono utili nelle applicazioni di software avanzato.

* in vendita separatamente

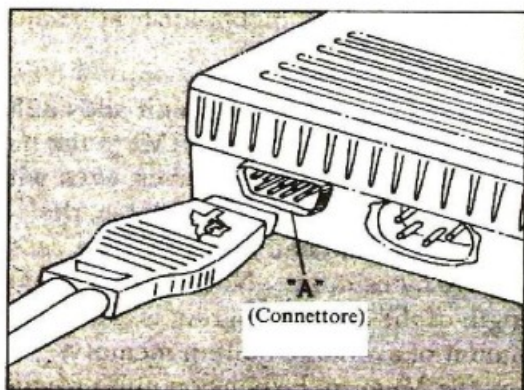
Il registratore: memoria esterna

Molti dei normali registratori a cassetta possono essere usati con il calcolatore, ma per operare nelle condizioni migliori devono avere le seguenti caratteristiche:

- Regolazione del volume
- Regolazione del tono
- Presa per il microfono
- Presa per controllo a distanza
- Cuffia o presa per uno speaker esterno
- Contatore digitale (Vi sarà utile per individuare le posizioni nel nastro di diversi programmi o insiemi di dati)

Nota - Il cavo per l'interfaccia con il registratore a cassetta finisce con tre spine per l'unità a cassette 1 "CS1", e con due spine per la 2 "CS2". L'unità a cassette 1 può essere usata sia per registrare che per leggere, l'unità 2 solo per registrare.

COLLEGAMENTO DEL REGISTRATORE
Per collegare il vostro registratore al calcolatore, usate solo il cavo TI per interfaccia con registratore, e procedete come segue:

1. Inserite il capo a spina singola con il connettore "D" a 9 pin nell'attacco a 9 pin che si trova dietro la console, indicato con A.

2. Inserite il capo a tre spine nel registratore come segue:

- Inserite la spina con il filo rosso nella presa per il microfono
- Inserite la spina con il filo nero nella presa per controllo distanza (notate che questa presa è più piccola delle altre due)
- Inserite la spina con il filo bianco nella presa per la cuffia (o per lo speaker esterno) - solo per CS1

3. Assicuratevi di come sono collegati i registratori CS1 e CS2 quando registrate dei dati. Ricordate che i dati possono essere caricati solo da CS1. Leggete il paragrafo SCRITTURA E LETTURA DATI per avere maggiori informazioni.

(*Nota* - Quando collegate solo un registratore, l'altro capo del connettore sarà inattivo.)

Quando avete collegato i cavi, regolate il tono del vostro registratore tra la metà e il massimo degli ACUTI o sulla posizione indicata sul suo manuale di istruzioni. Regolate il volume a circa metà della scala (se ha dieci posizioni, regolatelo su cinque, o sulla posizione indicata sul suo manuale). Se il vostro registratore non ha la regolazione del tono, può darsi che dobbiate regolare il volume al massimo, per avere i risultati migliori

SCRITTURA E LETTURA DATI
Se avete collegato il vostro registratore alla console come vi abbiamo detto, siete pronti a registrare e leggere dati.

Prima di farlo, assicuratevi che:

- I vostri nastri siano di alta qualità, per avere ottime prestazioni.
- Il nastro non sia più lungo del C-60. I nastri più lunghi non più sottili, e non garantiscono una buona fedeltà.
- Il registratore sia a più di mezzo metro dal video o dal televisore, per minimizzare interferenze di campo magnetico.
- Il nastro non venga mai avvicinato a meno di mezzo metro a un video, un televisore, un motore elettrico o qualsiasi altra sorgente di campo magnetico, per evitare che i vostri dati vengano danneggiati.
- Il sistema (console, registratore e video) non si trovi tutto su di una stessa superficie metallica, per minimizzare disturbi elettrici.
- Solo CS1 può essere usata per leggere; sia CS1 che CS2 possono essere usate per registrare.

Come registrare e leggere in TI BASIC

Le istruzioni complete per eseguire queste operazioni sono sotto il comando SAVE.

Il registratore: memoria esterna

Come registrare con un Modulo

Una volta inseriti i dati nel calcolatore e collegato il registratore (con una cassetta di buona qualità) potete registrare. Selezionate l'opzione "SAVE" del Modulo che state usando. Il calcolatore vi presenta un elenco di opzioni per la registrazione. (*Nota*-Se selezionate un'opzione che richiede un dispositivo non collegato e acceso, il calcolatore vi invia un messaggio di errore.) Supponiamo di voler registrare i dati sull'Unità a cassette 1. Selezionate CS1 dalla lista opzioni.

Da questo momento, il calcolatore vi guida nell'utilizzo della routine SAVE con le opportune istruzioni sul video. (Notate che le istruzioni sono le stesse per CS1 o CS2.) Il calcolatore controlla l'alimentazione del motore del registratore, e quindi il nastro non comincia a muoversi fin quando non premete **ENTER**.

Istruzioni sul video.

* REWIND CASSETTE TAPE CS1
THEN PRESS ENTER

Esecuzione.

Riavvolgete il nastro prima di premere **ENTER**. Se il vostro registratore *non* ha un indicatore della posizione del nastro, riavvolgete il nastro fino all'inizio. Se ce l'ha, posizionate il nastro nel punto in cui volete iniziare la registrazione e premete il tasto "stop" del registratore. (Prendete nota della posizione per poterci tornare in seguito.) Infine premete **ENTER** per continuare.

* PRESS CASSETTE RECORD CS1
THEN PRESS ENTER

Premete il tasto "record" sul registratore, e poi il tasto **ENTER** del calcolatore. Immediatamente ha inizio la registrazione su nastro, e sul video compare il messaggio

* RECORDING

Potete sentire il suono delle informazioni codificate, mentre sono registrate o lette nell'unità a nastro.

* PRESS CASSETTE STOP CS1
THEN PRESS ENTER

Quando avete finito di registrare, premete il tasto "stop" sul registratore e poi il tasto **ENTER** del calcolatore.

Il calcolatore vi chiederà

* CHECK TAPE (Y OR N)

Nota - Tutte le risposte alle domande che il calcolatore vi fa mentre registrate o leggete dati su cassetta devono essere costituite da una sola lettera maiuscola (Y, N, R, etc.). Tenete premuto il tasto SHIFT e premete la lettera opportuna.

A questo punto potete scegliere se fare eseguire al calcolatore un controllo sul nastro per assicurarsi che la registrazione sia stata fatta correttamente. Vi raccomandiamo vivamente di farlo per essere certi dell'affidabilità del vostro nastro per successivi usi *Nota*- Solo con CS1.

Se non volete controllare il nastro premete N. Rimuovete il nastro e attaccate un'etichetta per poterlo usare in seguito.

Se volete controllarlo, premete Y .Anche questa volta il calcolatore vi guida con i seguenti messaggi

* REWIND CASSETTE TAPE CS1
THEN PRESS ENTER

Riavvolgete il nastro *prima* di premere **ENTER**, fino al punto in cui avete cominciato a registrare. Se avete cominciato dall'inizio del nastro, riavvolgetelo tutto. Se invece avete cominciato in un altra posizione, riavvolgetelo fino a questo punto, e premete "stop" sul registratore. Infine premete **ENTER**.

* PRESS CASSETTE PLAY CS1
THEN PRESS ENTER

Premete il tasto "play" sul registratore, e poi **ENTER**. Il calcolatore confronterà i dati in memoria con quelli su nastro, e intanto visualizzerà il messaggio

* CHECKING

Se non ci sono errori, sul video compaiono i seguenti messaggi:

* DATA OK
PRESS CASSETTE STOP CS1
THEN PRESS ENTER

Il registratore: memoria esterna

Ora potete rimuovere il nastro e etichettarlo per usarlo in seguito.

Se invece c'è stato un errore di registrazione, riceverete uno dei seguenti due messaggi:

Messaggio di Errore

* ERROR - NO DATA FOUND

Significato

I vostri dati non sono stati registrati, o il nastro non può essere riletto.

PRESS R TO RECORD CS1
PRESS C TO CHECK
PRESS E TO EXIT

Messaggio di Errore

* ERROR IN DATA DETECTED

Significato

Una parte dei vostri dati non è stata registrata correttamente.

PRESS R TO RECORD CS1
PRESS C TO CHECK
PRESS E TO EXIT

Prima di proseguire, controllate ancora una volta che

■ Il registratore sia alla giusta distanza dal televisore (più di mezzo metro) ■ Il registratore sia collegato correttamente al calcolatore ■ La cassetta sia in buone condizioni (se avete dei dubbi, prendetene un'altra) ■ Il volume e il tono del registratore siano regolati correttamente, e in particolare che il volume non sia nè troppo alto nè troppo basso ■ La testina del registratore non abbia bisogno di essere pulita ■ Il sistema non sia appoggiato su di una superficie metallica.

Quando avete controllato tutto, potete scegliere fra una delle seguenti opzioni:

■ Premere R per registrare nuovamente i vostri dati, usando le istruzioni che abbiamo visto prima.
■ Premere C per dire al calcolatore di controllare un'altra volta i dati.
■ Premere E per "uscire". Comparirà il seguente messaggio

* PRESS CASSETTE STOP CS1
THEN PRESS ENTER

Il tasto E vi riporta all'inizio dell'opzione "Save" dei moduli, perciò quando premete **ENTER** potete memorizzare i vostri dati, seguendo le istruzioni che compaiono sullo schermo.

Come leggere con un Modulo

La prossima volta che volete usare i dati che avete registrato su nastro, dovete "caricarli" in memoria - cioè leggere i dati dal nastro *nella* memoria del calcolatore.

Nota-A causa delle differenze tra registratori, può darsi che un nastro registrato su di un modello non sia leggibile su un altro registratore.

Prima di tutto collegate il registratore al calcolatore. Poi inserite nel calcolatore il modulo da cui avete registrato le informazioni. Quando siete pronti a "caricare", selezionate l'opzione "LOAD DATA" del modulo. Quando il calcolatore ve lo chiede, premete 1 per indicare che le informazioni vanno lette da cassetta, e poi ancora 1 per selezionare l'unità a cassette CS1. Ricordatevi che per leggere dati si usa la CS1.

Da questo punto in avanti il calcolatore invia sul video le istruzioni per la lettura.

Istruzioni sul video

* REWIND CASSETTE TAPE CS1
THEN PRESS ENTER

Esecuzione

Riavvolgete il nastro *prima* di premere **ENTER**. Posizionate il nastro al punto da cui volete leggere i dati (all'inizio se il registratore non ha l'indicatore di posizione). Poi premete **ENTER**.

Il registratore: memoria esterna

Istruzioni sul video

* PRESS CASSETTE PLAY CS1
  THEN PRESS ENTER

Esecuzione

Premete il tasto "play" sul registratore e il tasto **ENTER** sul calcolatore. Le informazioni vengono lette dal nastro e caricate in memoria. Mentre il calcolatore legge il nastro, sullo schermo appare il messaggio

* READING

Ci vuole un pò di tempo a leggere i dati, a seconda della loro quantità.
Quando il calcolatore ha finito, vi comunica se ha letto correttamente o no. Se sì, sul video compariranno i seguenti messaggi.

* DATA OK
* PRESS CASSETTE STOP CS1
  THEN PRESS ENTER

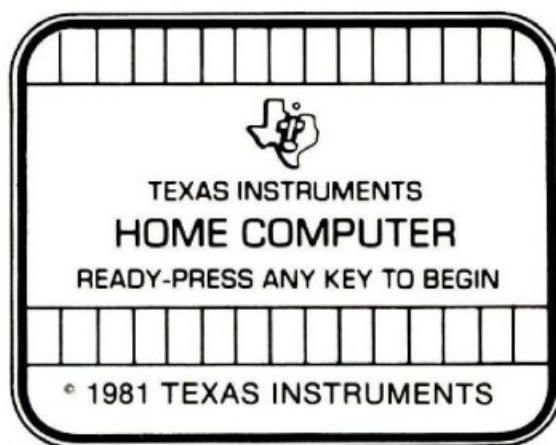
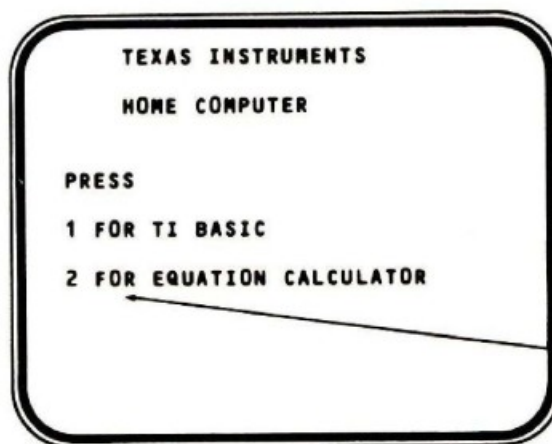
Ora potete cominciare a lavorare con il vostro modulo.
Se invece i dati non sono stati caricati correttamente, il calcolatore vi invia un messaggio di errore. Seguite le istruzioni sul video per riprovare a caricare i vostri dati correttamente.

Se incontrate ancora difficoltà, assicuratevi che:

- state usando il nastro giusto
- il nastro è posizionato correttamente da dove cominciano i dati
- il nastro non è stato danneggiato o cancellato
- il registratore è alla giusta distanza dal televisore (più di mezzo metro)
- il registratore è collegato correttamente al calcolatore
- il volume del registratore è regolato correttamente
- il sistema non è appoggiato su una superficie di metallo
- il nastro è stato registrato con la vostra unità a cassetta o con un modello identico
- la testina del registratore è pulita
- state usando l'unità CS1

Elementi di Programmazione

Questo capitolo vi mostra con degli esempi come è semplice comunicare con il vostro calcolatore, il che si rivela particolarmente utile per i principianti. Per poter realmente comunicare con il calcolatore, dovete imparare il suo linguaggio (TI BASIC), ma prima ancora dovete prendere confidenza con lui, imparando come lo si programma.

Quando siete pronti, accendete il calcolatore. Sul video comparirà la seguente figura:

Premete un tasto qualsiasi. Sul video comparirà la lista per la selezione.

(*Nota* - Quando volete "uscire" dal TI BASIC inserite la parola BYE e premete **ENTER**. Sul video tornerà il Titolo Principale.)

Gli esempi di questo libro sono stampati con lettere maiuscole. Se volete riprodurli esattamente come li vedete qui, premete il tasto **ALPHA LOCK**. In genere, comunque, il calcolatore accetta sia le maiuscole che le minuscole.

Ora premete 1 per selezionare TI BASIC. Il video vi segnala che il calcolatore è pronto a ricevere le vostre istruzioni.

Elementi di Programmazione

Modo Immediato (o di Comando)

In Modo Immediato il calcolatore esegue "immediatamente" ogni istruzione BASIC che scrivete, non appena premete **ENTER**. Poichè la risposta compare immediatamente sul video, il Modo Immediato è un buon modo per provare e conoscere alcune istruzioni del BASIC.

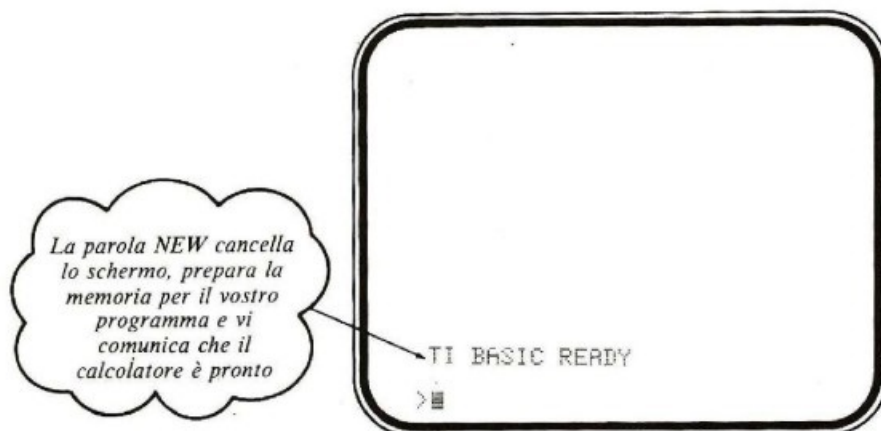
Con le istruzioni in Modo Immediato il calcolatore esegue un'operazione alla volta. Ogni istruzione è eseguita immediatamente dopo che avete premuto il tasto **ENTER**.
Per esempio, se scrivete

PRINT "ECCOCI QUA"

e premete **ENTER**, il calcolatore stampa ECCOCI QUA

Ora siete in grado di affrontare i *programmi*, cioè insiemi di istruzioni che *non* vengono eseguite immediatamente, ma che vengono conservate nella memoria del calcolatore, in attesa che voi diate il comando di eseguirle.

Un Programma di Stampa

Cominciamo a usare l'istruzione PRINT in un programma. Inserite la parola NEW e premete **ENTER**.

Ora scrivete il seguente programma, premendo **ENTER** alla fine di ogni linea di programma

Mentre scrivete il programma, notate il simbolo di "pronto" (prompt) che compare alla sinistra dell'area di stampa. Questo simbolo segna l'inizio di ogni linea di programma.

Nella terminologia dei calcolatori si dice che avete "inserito" un programma. Tutto qui. Ora controllate il programma per vedere se avete commesso errori di battitura. Se ce ne sono, riscrivete la linea correttamente, *compreso il numero che si trova all'inizio della linea*, in basso sul video. Poi premete **ENTER**. Il calcolatore sostituirà automaticamente la linea vecchia con quella nuova corretta.

Quando siete pronti ad eseguire il programma, scrivete CALL CLEAR e premete **ENTER**. Il video sarà cancellato, ma il vostro programma resta nella memoria del calcolatore.

Battete RUN e premete ancora **ENTER**.

Se volete eseguire un'altra volta il programma, premete ancora RUN e **ENTER**

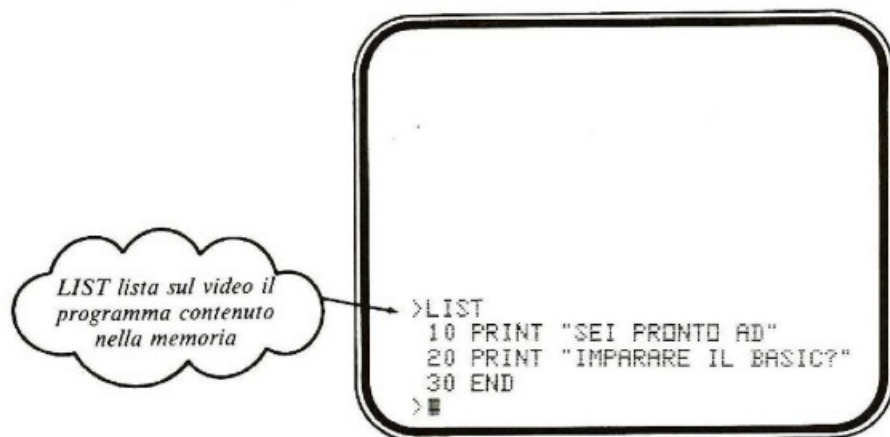
Ogni volta che premete RUN e **ENTER**, il calcolatore comincia dalla prima istruzione ed esegue nell'ordine le istruzioni fino all'ultima. END significa letteralmente fine.

Avete notato che il video diventa verde per qualche attimo mentre il programma è in esecuzione? Il video diventa sempre verde durante l'esecuzione di un programma, e torna al suo abituale colore blu alla fine.

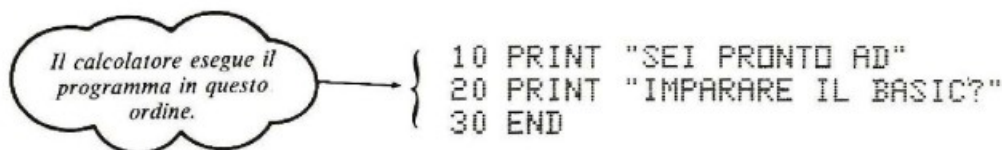
Struttura di un Programma

Ora che avete un minimo di esperienza di programmazione, ripassiamo alcune delle operazioni che avete eseguito quando avete inserito il programma precedente. Per rinfrescarvi la memoria, riporterete il programma sul video.

Elementi di Programmazione

Scrivete CALL CLEAR (*senza* un numero di linea) e premete **ENTER** per cancellare il video. Ora battete LIST e ancora **ENTER**

Il programma precedente consiste di tre istruzioni o "linee". Ogni istruzione comincia con un *numero di linea*, che ha due importanti funzioni

1. Comunica al calcolatore di non eseguire immediatamente l'istruzione, ma di conservarla in memoria quando premete **ENTER**.
2. Determina l'ordine con cui le istruzioni devono essere eseguite nel programma.

Come in Modo Immediato, avete premuto **ENTER** dopo ogni linea del programma. **ENTER** definisce la fine di una linea di programma, così come il numero di linea ne definisce l'inizio, e dice al calcolatore di registrare la linea in memoria. *La parola* **ENTER** *è essenziale alla fine di ogni linea*; senza di essa, la linea non viene memorizzata correttamente.

Vi chiederete perchè abbiamo numerato le linee di 10 in 10 (10,20,30,etc.). Avremmo benissimo potuto numerarle 1,2,3, ma usando un incremento di 10, o di 100 come 100, 200, 300 etc. abbiamo la possibilità di inserire nuove linee se vogliamo espandere il programma, senza doverlo riscrivere completamente. (Capirete meglio questo espediente quando parleremo di modifiche e correzioni di un programma.)

Comandi: NEW, RUN, LIST

Avete già usato questi comandi, ma vi sarà utile qualche informazione più precisa sui comandi in generale, e in particolare su: NEW, RUN, LIST.

I *comandi* sono una cosa diversa dalle *istruzioni*. Non fanno parte di un programma, e non hanno un numero di linea. Dicono al calcolatore di eseguire operazioni particolari.

NEW - Dice al calcolatore di cancellare il programma in memoria. (Cancella anche il video, ma *non è* da confondere con la CALL CLEAR, che cancella *solo* il video.)

RUN - Dice al calcolatore di eseguire ("run") il programma che si trova in memoria.

LIST - Dice al calcolatore di visualizzare (o "listare") sul video il programma che si trova in memoria.

Come abbiamo visto prima, usiamo NEW *solo* quando vogliamo predisporre il calcolatore a memorizzare un nuovo programma. Usate NEW con attenzione. Se siete in dubbio, usate LIST per vedere il programma in memoria prima di cancellarlo.

Avete già imparato a cosa serve il comando RUN. E' una parola magica che fa accadere di tutto.

Un Programma di Calcolo

Oltre alla capacità di stampare e di inviare messaggi, il calcolatore ha la capacità di fare calcoli. I tasti che si usano per eseguire le quattro operazioni matematiche sono

SHIFT + per le addizioni
SHIFT - per le sottrazioni
SHIFT * per le moltiplicazioni
/ per le divisioni

Proviamo a scrivere un programma di calcolo.
Come esempio possiamo scrivere un programma che converte i kilogrammi in libbre (1 kilogrammo = 2.2 libbre). Per prima cosa puliremo il video e la memoria battendo NEW e **ENTER**. Useremo le variabili K (per i kilogrammi) e P (per le libbre, cioè Pound) per ricordarci che valore contengono, e cominceremo il nostro programma assegnando loro dei valori.

Inserite

Vogliamo determinare a quante libbre equivalgono 50 kilogrammi, e quindi abbiamo assegnato a K il valore 50. Notate che abbiamo definito P=2.2*K. Se ci fermassimo qui ed eseguissimo il programma, il calcolatore eseguirebbe i calcoli, ma non ci mostrerebbe il risultato. Aggiungete

```
30 PRINT P
```

e premete **ENTER**. Abbiamo detto al calcolatore tutto quello che deve fare. Gli abbiamo detto il numero di kilogrammi che vogliamo convertire in libbre, come eseguire la conversione e come comunicarci la risposta. È tutto quello di cui abbiamo bisogno, perciò scrivete

```
40 END
```

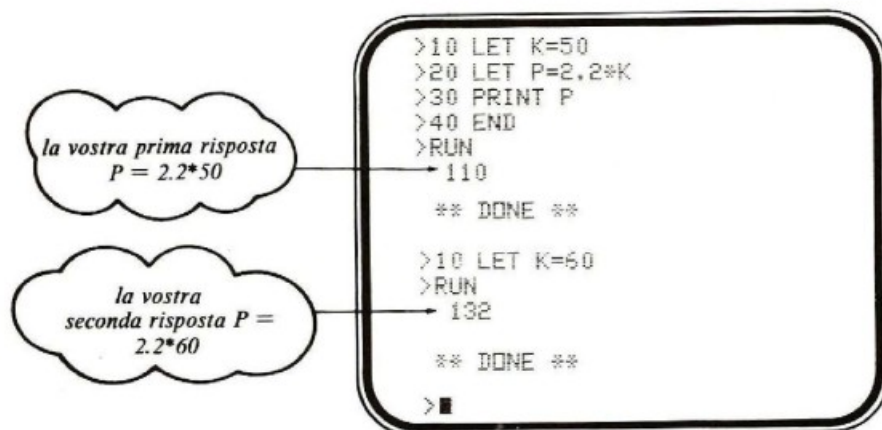
e premete **ENTER**. Il vostro programma si presenterà come nella figura seguente

Prima di eseguire il programma, segnaliamo due caratteristiche del TI BASIC un po' diverse dalle altre versioni di questo linguaggio. Prima di tutto c'è un "prompt"(a sinistra dell'area di stampa del video)che segna l'inizio di ogni linea di programma. Capirete meglio la sua funzione quando inserirete linee di programma più lunghe di una riga del video. Inoltre l'istruzione END è opzionale in TI BASIC, ma poichè è un elemento convenzionale del BASIC la useremo in questo esempio.

Ora controllate che il programma non abbia errori di battitura. Se ve ne sono, riscrivete la linea correttamente, compreso il numero di linea, e premete **ENTER**. Quando siete pronti, battete RUN e **ENTER**.

La risposta è sul video: 50 kilogrammi equivalgono a 110 libbre. Supponiamo ora di voler sapere a quante libbre equivalgono 60 kilogrammi. Facile, basta cambiare la linea 10. Battete

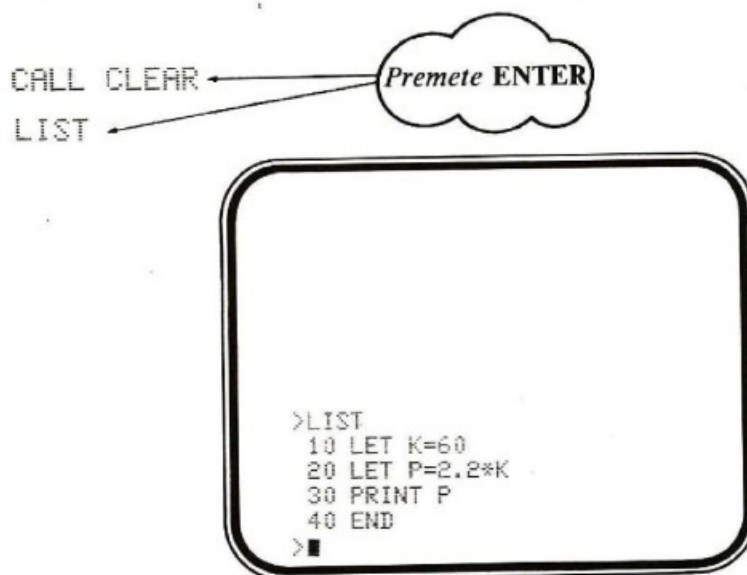
```
10 LET K=60
```

e premete **ENTER**. Ora eseguite il programma un'altra volta.

Editing (modifiche) del Programma

Avete fatto quello che si chiama un "editing" del programma. La possibilità di modificare un programma senza ribatterlo tutto è una di quelle cose che apprezzerete sempre di più man mano che aumenta la vostra abilità nel programmare. Per avere un'idea dei suoi vantaggi, proviamola per fare alcune ulteriori modifiche al programma precedente.

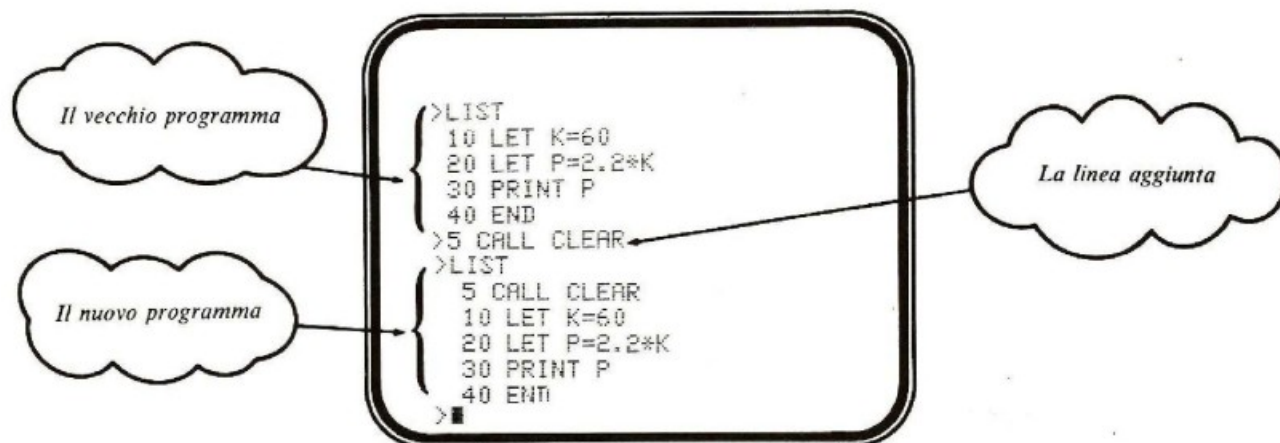
Come aggiungere linee ad un programma

Abbiamo visto prima che è meglio numerare le linee di un programma di 10 in 10 (invece che 1,2,3,etc.) per consentire l'aggiunta di linee senza dovere riscrivere tutto il programma.

Battete

(Notate che il prompt non compare sulla sinistra delle linee stampate dal calcolatore, ma solo su quelle inserite da voi.)

Possiamo aggiungere un'istruzione CALL CLEAR al programma, per non dover cancellare il video con un comando da tastiera ogni volta che lo eseguiamo.

Battete 5 CALL CLEAR ← *Premete* **ENTER**

Ora listate ancora il programma per vedere la nuova linea (battete **LIST** e premete **ENTER**).

Confrontate i due programmi sul video, e notate che il calcolatore ha sostituito automaticamente la nuova linea al posto giusto. Eseguite un'altra volta il programma per vedere l'effetto della nuova linea.

Ora aggiungiamo un'altra linea che ci servirà a evidenziare meglio la nostra risposta. Battete

```
27 PRINT "RISPOSTA:"
```

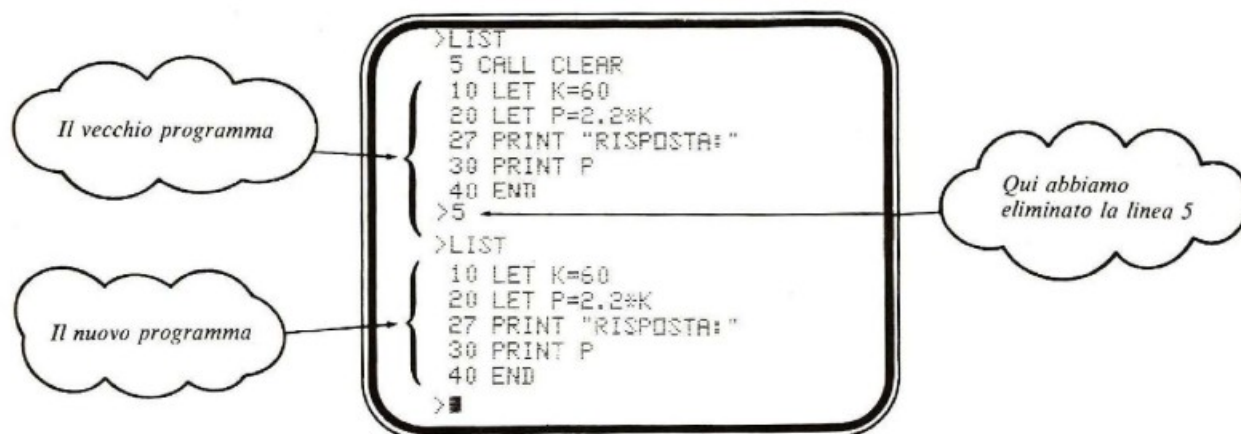
e premete **ENTER**. Quando eseguite ancora il programma, vedrete

Come togliere linee da un programma

Molto spesso è necessario eliminare linee da un programma. E' molto semplice.

Nel programma che abbiamo appena scritto non ci sono, in realtà, linee da eliminare, ma per fare un esempio cancelliamo la linea 5.

Cancelliamo il video e listiamo il programma così come è adesso. La linea 5 è la prima del programma, e contiene l'istruzione CALL CLEAR. Per eliminarla battete 5 e premete **ENTER.**

Ora listate ancora il programma . La linea 5 è scomparsa.

E' tutto qua. Per eliminare una linea, battete il numero di linea e premete **ENTER.** Il calcolatore cancellerà la linea dalla memoria del calcolatore.

Poichè in realtà abbiamo bisogno della linea 5 nel programma, la rimetteremo al suo posto. Battete

```
5 CALL CLEAR
```

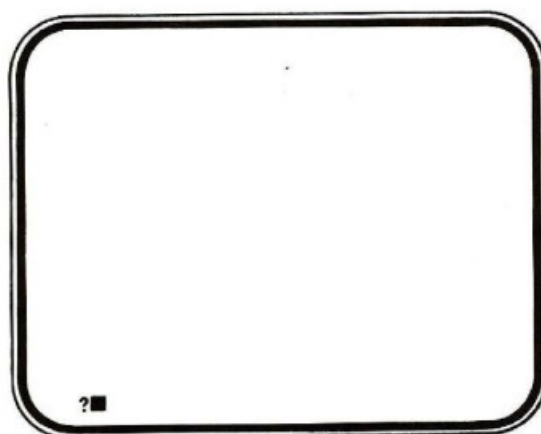
e premete **ENTER.**

L'istruzione INPUT

Avete visto che potete cambiare facilmente il valore di K modificando la linea 10 per dargli un nuovo valore. Supponiamo di avere molti valori per K, e di voler trovare il corrispondente valore di P per ognuno di essi. Sarebbe scomodo riscrivere la linea 10 ogni volta.

E' meglio modificare la linea 10. Un'istruzione di INPUT fa sì che il calcolatore stampi un "?" e si fermi, in attesa che voi inseriate un valore e premiate **ENTER**. Il valore che inserite viene assegnato alla variabile che compare nell'istruzione di INPUT. Per esempio, battete

```
10 INPUT K
```

e premete **ENTER**. Ora eseguite il programma un'altra volta.

Il "?" e il cursore vi segnalano che il calcolatore aspetta che voi inseriate un valore per K. Questa volta poniamo K = 70, quindi inserite *70* e premete **ENTER**. Il calcolatore stampa la risposta

Ora potete eseguire il programma tutte le volte che volete, cambiando il valore di K ogni volta che il calcolatore stampa un "?" e si ferma.

L'istruzione INPUT può essere usata per stampare un messaggio di richiesta (invece che un "?") per ricordarvi la natura del valore che dovete inserire.
Modificate la linea 10 come segue

e premete **ENTER**. Ora eseguite il programma un'altra volta. Vi chiederà

```
CHILOGRAMMI?
```

Questa volta ponete K = 50. Battete 50 e premete **ENTER**.

A questo punto il programma si presenta così

```
5 CALL CLEAR
10 INPUT "CHILOGRAMMI?":K
20 LET P=2.2*K
27 PRINT "RISPOSTA:"
30 PRINT P
40 END
```

Se volete, potete listarlo sul video e rivedere le modifiche che avete fatto. Quando siete pronti, procederemo ad altre modifiche.

Variabili alfanumeriche (stringhe)

Sapete già che cosa sono le *varianti numeriche*: valori numerici assegnati a dei nomi (variabili), come "K = 50". Una variabile alfanumerica è una combinazione di caratteri (lettere o numeri, o altri simboli) assegnata a un nome. Le variabili alfanumeriche differiscono da quelle numeriche in quanto

1. Il nome della variabile deve finire con un $.
2. I caratteri alfanumerici della stringa devono essere tra virgolette.
3. Non si possono eseguire operazioni aritmetiche su stringhe di numeri.

Facciamo qualche esempio di istruzioni in Modo Immediato prima di cambiare il programma. (Notate che ciò non interferisce con il programma in memoria).

Cancellate il video (CALL CLEAR) e inserite

```
LET N$="BUONA·SERA"
PRINT N$
```

Ora inserite

```
LET W$=" GIORGIO"
PRINT N$;W$
```

Potete personalizzare il vostro programma di conversione usando una variabile alfanumerica (o di stringa). Scrivete le seguenti linee:

(Cancellate il video e listate nuovamente il programma, per vedere dove sono state inserite le due linee).

Quando eseguite il programma le due istruzioni di INPUT faranno fermare il programma due volte.

Il calcolatore vi chiede	*Voi rispondete*
NOME, PER PIACERE?	Inserendo il vostro nome e premendo **ENTER**.
CHILOGRAMMI?	Inserendo il numero di kilogrammi e premendo **ENTER**

Provate. Battete RUN e premete **ENTER**.

Noi inseriremo PIERO e **ENTER**, quindi vedremo

Inserite ancora 70 per il numero di kilogrammi, poi premete **ENTER** e vedrete

Le variabili di stringa vi risparmiano un gran lavoro di battitura, quando usate un messaggio molte volte in un programma.

Ora listate il programma per rivedere le ultime modifiche. Vi abbiamo dato un gran numero di informazioni, in poco tempo. Adesso potete fare degli esperimenti da soli, mettendo in pratica alcune delle cose che avete imparato.

Provate

Provate a scrivere un altro programma di conversione, da gradi Fahrenheit (F) a gradi Celsius (C). La formula di conversione è

Gradi C = 5/9 (gradi F - 32)

Non dimenticate di usare le istruzioni INPUT e CALL CLEAR al momento giusto. Vi diamo un suggerimento - ponete C = 5/9 * (F - 32) con le parentesi così come sono nella formula.

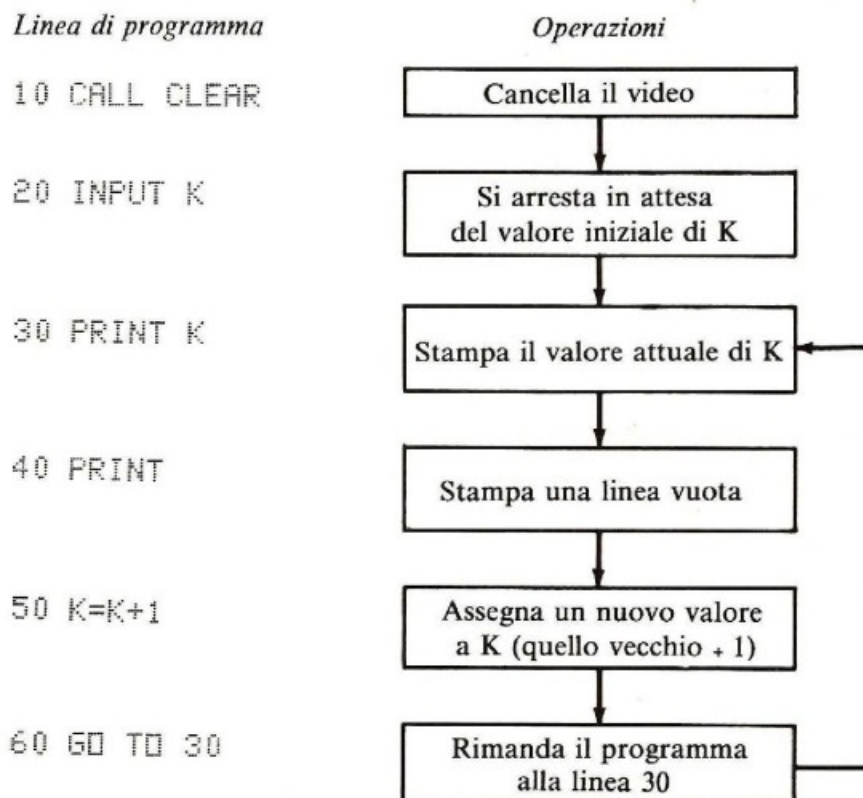
L'istruzione GO TO

I programmi che abbiamo scritto finora sono eseguiti dall'inizio alla fine in un ordine strettamente sequenziale. Ci sono dei casi in cui questo ordine deve essere modificato. Leggete il programma seguente, ma per ora non inseritelo nel calcolatore.

```
10 CALL CLEAR
20 INPUT K
30 PRINT K
40 PRINT
50 K=K+1
60 GO TO 30
```

Con l'istruzione GO TO della linea 60 "rimandiamo" il programma indietro alla linea 30, in modo che le operazioni indicate nelle linee 30, 40 e 50 siano ripetute più volte, creando quello che si chiama un *ciclo* (o *loop*). Notate che non abbiamo usato l'istruzione END, visto che il programma non arriverà mai oltre la linea 60. Il programma non avrà termine, fin quando non lo interromperete voi, premendo **CLEAR**. Questo si chiama "ciclo senza fine".

Inserite il programma. Battete NEW e premete **ENTER** per cancellare la memoria, poi scrivete le linee seguenti

Prima di eseguire il programma esamineremo un diagramma a blocchi (*flowchart*), che spiega come funziona.

Ora eseguite il programma, assegnando 1 a K come valore iniziale. Osservate con che rapidità il calcolatore esegue i calcoli, quasi troppa per poterli seguire. Per questo abbiamo aggiunto la linea vuota (linea 40), per separare un po' i numeri tra di loro, in modo da vederli meglio.

Lasciate che il calcolatore continui a eseguire calcoli, quando volete interromperlo, premete **CLEAR**. Sul video comparirà *BREAKPOINT AT (#) con il numero dell'istruzione a cui si è fermato il programma. Eseguite il programma tutte le volte che volete, usando ogni volta numeri diversi come valore iniziale di K (50, 100, 500, etc.).

Potete scrivere indifferentemente GO TO o GOTO. Il calcolatore non ci fa caso. Se però cercate di rimandare il programma ad un numero di linea che non esiste, vi invia un messaggio di errore.

Supponiamo per esempio di scrivere

```
60 GO TO 25
```

e premere **ENTER**. Provate, eseguite il programma e state a vedere cosa succede. Vedrete il seguente messaggio di errore:

```
* BAD LINE NUMBER IN 60
```

Correggete la linea così:

```
60 GO TO 30
```

ed eseguite ancora il programma.

Possiamo cambiare il programma per fargli calcolare i risultati corrispondenti a valori di K che differiscono di 2 o di 5?

Certo, basta una piccola modifica. Scrivete

```
50 K=K+2
```

e premete **ENTER**. Ora eseguite il programma, assegnando 2 come valore iniziale di K.

Provate ancora con incrementi di 3, 5, 10, ecc.

Comandi e Istruzioni TI BASIC

Introduzione

Questo paragrafo fornisce una spiegazione completa di tutti i comandi e le istruzioni di TI BASIC. Come abbiamo già detto, il BASIC è un linguaggio facile da usare anche per principianti, ma abbastanza potente da consentirvi una gran quantità di applicazioni. Ci sono tre strade da seguire per imparare senza difficoltà il TI BASIC.

Se siete un principiante che vuole acquisire maggiore familiarità con il calcolatore, vi consigliamo di studiare il libro "BASIC for beginners" della Texas Instruments. In modo rapido e divertente vi fa acquisire una prima esperienza di programmazione con una tecnica di auto-apprendimento.

Se avete già una certa esperienza di programmazione, e volete diventare più esperti di TI BASIC e di come funziona sul vostro calcolatore, vi forniamo una serie di programmi applicativi alla fine di questo manuale. Questi programmi partono da un livello molto semplice, e diventano gradualmente più complessi. Studiandoli imparerete l'uso di molte istruzioni di TI BASIC; nel manuale ci sono poi gli approfondimenti necessari.

Per coloro che hanno una certa esperienza di programmazione, ma non di BASIC, o che vogliono rinfrescarsi la memoria, consigliamo un ottimo libro di Herbert Peckham, *Programming BASIC with the TI Home Computer*, che dà rapidamente un'esperienza di alto livello in BASIC. E' in vendita in qualsiasi libreria ben fornita.

Per i più esperti , che hanno un'ottima conoscenza della programmazione, questa guida può servire come riferimento per le istruzioni e i comandi di TI BASIC, fornendo i dettagli che vanno rinfrescati ogni tanto. TI BASIC è in linea con le convenzioni American National Standard per il Minimal BASIC. Inoltre in questo manuale sono descritte le altre prestazioni del TI BASIC, come la grafica a colori, la musica etc. Se siete un esperto programmatore BASIC, non avrete problemi a impadronirvi del TI BASIC.

Comandi e Istruzioni TI BASIC

Organizzazione del manuale

Questo manuale è suddiviso in modo da permettere una facile lettura e consultazione.

1. Informazioni di carattere generale
2. Comandi
3. Istruzioni di assegnazione e di controllo
4. Istruzioni di Ingresso/Uscita
5. Grafica a colori e suoni
6. Funzioni aritmetiche
7. Funzioni di stringa
8. Funzioni definite dall'utente
9. Matrici
10. Sottoprogrammi
11. Gestione file

Alla fine del manuale c'è un glossario con i termini più usati.

Notazioni convenzionali

All'inizio della descrizione di ogni comando o istruzione TI BASIC, compare una linea che mostra il suo formato. In queste linee sono state usate delle notazioni convenzionali, che descriviamo qui per aiutarvi a usarle.

{ } - Le parentesi graffe indicano che potete scegliere solo uno degli elementi tra parentesi.

[ ] - Le parentesi quadre indicano un elemento opzionale; potete usarlo, ma non è necessario.

... - I puntini indicano che l'elemento precedente può essere ripetuto tante volte quante volete.

MAIUSCOLE - Se usate parole con lettere maiuscole dovete inserirle esattamente come vi dicono le istruzioni.

corsivo - Le parole in corsivo sono una descrizione degli elementi che devono comparire in quella posizione. Quando scrivete il comando o l'istruzione dovete sostituire alle parole in corsivo le vostre informazioni.

Esempi

Per ogni istruzione o comando, sono mostrati sulla destra esempi di programmi. Ogni linea che dovete inserire è indicata con il carattere di prompt (>) a sinistra, proprio come appare sul video.

Informazioni di carattere generale

Introduzione

Una volta installato il calcolatore, è semplice usare il TI BASIC. Quando accendete il calcolatore, sul video compare il Titolo Principale. Premete un tasto qualsiasi per visualizzare la lista di selezione. Quando appare, premete 1 per selezionare il TI BASIC. Sul video ci sono solo le parole "TI BASIC READY" e un prompt (>) seguito dal cursore intermittente (■). Quando il cursore è sul video, il calcolatore sta aspettando che voi inseriate qualcosa. Il prompt segna l'inizio di ogni linea che inserite.

Ogni linea del video, contiene fino a 28 caratteri. Le istruzioni e i comandi possono essere lunghi al massimo quattro linee del video. Quando avete riempito una linea, il cursore passa automaticamente alla successiva mentre voi continuate a scrivere. Quando avete riempito quattro linee, il calcolatore accetta ancora caratteri, ma il cursore non si sposta. Ogni carattere che battete va a sostituire l'ultimo carattere della linea.

Tutti i tasti del paragrafo Tasti Speciali possono essere usati per correggere le linee del programma prima di premere **ENTER**. Per cambiare una linea dopo che avete premuto **ENTER**, potete riscrivere la linea con le correzioni necessarie, oppure entrare in "Modo Edit". Notate che quando fate delle modifiche ad un programma, tutti i file aperti sono chiusi (vedi l'istruzione OPEN) e tutte le variabili diventano indefinite.

Tasti Speciali

Molti tasti hanno funzioni speciali in TI BASIC.

ENTER - quando lo premete , il calcolatore memorizza l'ultima linea di programma che avete scritto.

FCTN = (QUIT) - Fa tornare sul video il Titolo Principale. Quando il calcolatore esce dal TI BASIC, i programmi e i dati vengono cancellati dalla memoria.

FCTN ↑ (UP) - Ha la stessa funzione del tasto **ENTER**, tranne che in Modo "Edit".

FCTN ↓ (DOWN) - Idem come sopra.

FCTN ← (LEFT) - Sposta il cursore di una posizione a sinistra ogni volta che viene premuto. Quando il cursore passa su di un carattere, non lo cancella nè lo modifica in alcun modo. Quando arriva all'inizio della linea, questo tasto non ha più effetto.

FCTN → (RIGHT) - Sposta il cursore di una posizione a destra ogni volta che viene premuto. Anche in questo caso il cursore non modifica i caratteri su cui passa. Quando raggiunge la fine della linea (4 linee del video), il tasto non ha più effetto.

FCTN 2 (INS) - Serve per inserire caratteri all'interno di una linea di programma. Posizionate il cursore (usando **FCTN** ← o **FCTN** →) sul carattere immediatamente a destra della posizione in cui volete inserire i caratteri, e premete questo tasto. Da questo momento, ogni volta che battete un carattere il cursore e tutti i caratteri della linea che si trovano alla sua destra, sono spostati di una posizione a destra. Il carattere che avete battuto è inserito nella posizione che si libera. Notate che i caratteri che escono dalla linea del programma vengono cancellati.

FCTN 1 (DEL) - Serve per cancellare caratteri da una linea. Posizionate il cursore (usando **FCTN** ← o **FCTN** →) sul carattere che volete cancellare e premete il tasto. Il carattere viene cancellato e tutti i caratteri della linea a destra del cursore sono spostati a sinistra di una posizione.

FCTN 4 (CLEAR) - Questo tasto ha due funzioni, a seconda di quando lo usate.

Premendolo durante l'esecuzione di un programma, lo si fa fermare alla linea immediatamente successiva a quella in esecuzione. Il tasto va tenuto premuto fino a che il programma si ferma. Sul video comparirà il messaggio "BREAKPOINT AT numero di linea"; l'istruzione che si trova a quel numero di linea non è stata eseguita. Potete far ripartire il programma con il comando CONTINUE.

Se lo premete mentre state inserendo una linea di un programma, la linea non viene memorizzata. Questo tasto ha altre funzioni in Modo "Edit" e in Modo "Number".

Tasti Speciali

FCTN 3 (ERASE) - Cancella tutta la linea di programma che state scrivendo.

BARRA SPAZIATRICE - Muove il cursore di una posizione a destra ogni volta che viene premuta. Se spostate il cursore su di un carattere con questo tasto, il carattere viene cancellato e sostituito con uno spazio.

Blank (Spazi)

In genere ci possono essere spazi in qualsiasi parte del programma, senza influenzare l'esecuzione del programma stesso. D'altra parte, ogni spazio non necessario viene cancellato quando la linea del programma viene visualizzata con uno dei comandi EDIT, NUM o LIST. Ci sono però delle posizioni in cui non devono apparire spazi, e precisamente:

(1) in un numero di linea
(2) in una parola riservata
(3) in una costante numerica
(4) in un nome di variabile.

Seguono alcuni esempi di uso scorretto degli spazi; la linea corretta è riportata nella colonna a destra.

```
1 00 PRINT "HELLO"           100 PRINT "HELLO"
110 PR INT "COME STAI?"      110 PRINT "COME STAI?"
120 LET A=1 00               120 LET A=100
130 LET CO ST=24.95          130 LET COST=24.95
```

Tutte le parole riservate di un programma devono essere precedute e seguite immediatamente da uno dei seguenti caratteri:

- uno spazio
- un operatore aritmetico (+ - * / ∧)
- un operatore di stringa (&)
- un carattere speciale (< = > ( ) , ; :#)
- un carattere di fine linea (il tasto **ENTER**)

Numeri di Linea

Un programma è costituito da una sequenza di linee ordinate secondo un numero di linea che serve come etichetta. Ogni linea comincia con un numero intero tra 1 e 32767 compresi. Possono esserci degli zeri a sinistra del numero, ma vengono ignorati dal calcolatore. Per esempio: 033 e 33 sono letti entrambi come 33. Non è necessario inserire le linee in ordine, perchè vengono ordinate automaticamente dal calcolatore.

Quando eseguite il programma, le linee vengono eseguite in ordine sequenziale finchè

(1) si incontra un'istruzione di salto (vedi il paragrafo "Istruzioni di assegnazione e di controllo")
(2) si verifica un errore che interrompe l'esecuzione (vedi il paragrafo "Messaggi di errore")
(3) l'utente interrompe il programma con un comando di BREAK o con il tasto **CLEAR**
(4) viene eseguita un'istruzione di STOP o di END
(5) viene eseguita l'istruzione con il numero di linea massimo

Se scrivete una linea con un numero minore di 1 o maggiore di 32767, compare il messaggio "BAD LINE NUMBER" e la linea non viene memorizzata.

```
>0 A=2

 * BAD LINE NUMBER

>33000 C=4

 * BAD LINE NUMBER
```

Costanti Numeriche

Le costanti numeriche possono essere numeri reali positivi o negativi, con un numero qualsiasi di cifre. I numeri sono conservati all'interno in 7 digit in base 100, cioè con 13 o 14 cifre decimali, a seconda del loro valore.

Esempi:

Notazione Scientifica

Con la notazione scientifica si possono manipolare facilmente numeri molto piccoli o molto grandi. Un numero in notazione scientifica è espresso come una base (mantissa) per 10 elevato a una certa potenza (esponente).

```
>PRINT 1.2
 1.2
>PRINT -3
-3
>PRINT 0
 0
```

$$\text{Numero} = \text{Mantissa} \times 10^{\text{Esponente}}$$

Per inserire un numero con la notazione scientifica

- Inserite la mantissa (con il segno meno se è negativa)
- Battete la lettera "E" (E maiuscola)
- Inserite la potenza del 10 (col segno meno se è negativa)

Seguono alcuni esempi di numeri in notazione scientifica

Numero	*Inserito come*
3.264×10^{4}	3.264E4
-98.77×10^{21}	—98.77E21 o —9.877E22
5.691×10^{-5}	5.691E-5
-2.47×10^{-17}	—2.47E-17

```
>PRINT 3.264E4
 32640
>PRINT -98.77E21
-9.877E+22
```

Le costanti numeriche possono avere valori compresi tra -9.9999999999999E127 e -1E-128, e tra 1E-128 e 9.9999999999999E127, oltre naturalmente lo 0.

```
>PRINT 0
 0
```

Underflow - Se un numero, dato in ingresso o calcolato, una volta arrotondato è maggiore di -1E-128 e minore di 1E-128, si ha un "underflow". In questo caso il calcolatore sostituisce il valore del numero con zero, e continua l'esecuzione del programma. Non viene segnalato errore.

```
>PRINT -9E-130
 0
>PRINT 9E-142
 0
```

Overflow - Se il numero arrotondato ha un valore maggiore di 9.9999999999999E127 o minore di -9.9999999999999E127, si ha un "overflow". Il numero è sostituito con uno degli estremi previsti dal calcolatore, viene inviato un messaggio di avvertimento "NUMBER TOO BIG", e il programma continua. Gli estremi accettati dal calcolatore sono -9.9999999999999E127 o 9.9999999999999E127. Se l'esponente è maggiore di 99 viene stampato "**".

```
>PRINT 97E136
 * WARNING:
   NUMBER TOO BIG
 9.99999E+**
>PRINT -108E144
 * WARNING
   NUMBER TOO BIG
-9.99999E+**
```

Costanti alfanumeriche

Una costante alfanumerica (o costante di stringa) è una stringa di caratteri (comprese lettere, numeri, spazi, simboli, etc.) fra virgolette. Gli spazi in una costante di stringa non sono ignorati, ma sono contati come caratteri. Possono essere usati in una stringa tutti i caratteri della tastiera che possono essere visualizzati.Una stringa può essere lunga come una linea di ingresso (112 caratteri o quattro linee del video).

Esempi:

```
>NEW
>100 PRINT "ATTENZIONE:"
>110 PRINT "QUESTA E' UNA STRINGA
"
>120 PRINT "TUTTI I CARATTERI (+
- / @ ,) POSSONO ESSERE USATI"
>130 END
>RUN
 ATTENZIONE:
 QUESTA E' UNA STRINGA
 TUTTI I CARATTERI (+ - / @ ,)

 ) POSSONO ESSERE USATI

   ** DONE **
```

Quando viene eseguita un'istruzione PRINT o DISPLAY, le virgolette non sono visualizzate. Se volete che le parole o le frasi di una stringa siano stampate con le virgolette, dovete inserire una coppia di virgolette consecutive da entrambe le parti della parola o della frase.

```
>NEW
>100 PRINT "PER STAMPARE ""GLI A
PICI"" DOVETE USARE 2 VOLTE GLI
APICI"
>110 PRINT
>120 PRINT " TOM DISSE: ""AHI, A
HI"""
>130 END
>RUN
 PER STAMPARE "GLI APICI" DOVETE
 USARE 2 VOLTE GLI APICI

TOM DISSE: "AHI, AHI"

 ** DONE **
```

Espressioni Alfanumeriche

Le espressioni alfanumeriche sono costituite da variabili di stringa, costanti di stringa e riferimenti a funzioni, concatenate dall'operatore "&". L'operazione di concatenamento vi consente di collegare insieme diverse stringhe. Tutte le funzioni richiamate in un'espressione di stringa devono essere funzioni di TI BASIC (vedi il paragrafo Funzioni di Stringa) o funzioni definite con un'istruzione DEF, di tipo alfanumerico. Se il risultato di un'espressione di stringa ha un numero di caratteri superiore a 255, viene troncato sulla destra, e il programma prosegue. Non viene dato nessun avviso.

```
>100 A$="BELLA"
>110 B$="BRUTTA SERATA."
>120 C$="COME STAI?"
>130 MSG$=A$&SEG$(B$,7,7)
>140 PRINT MSG$&" "&C$
>150 END
>RUN
 BELLA SERATA COME STAI?

 ** DONE **
```

Notate che tutti i caratteri compresi in un'espressione di stringa sono visualizzati sul video esattamente come li avete battuti.

Variabili

In BASIC ogni variabile ha un nome, di uno o più caratteri, di cui il primo deve essere una lettera, un segno "at" (@), una parentesi quadra aperta ([), una parentesi quadra chiusa (]), una slash (\), o un trattino (-). I caratteri consentiti in un nome di variabile sono lettere, numeri, il segno "at" (@) e il trattino (-). Fa eccezione il dollaro ($). L'ultimo carattere di un nome di variabile alfanumerica *deve* essere un $, e questo non può comparire in nessun altra posizione in un nome di variabile. I nomi di variabile possono avere fino a 15 caratteri, compreso il dollaro per le variabili alfanumeriche.

I nomi di matrici seguono le stesse regole di quelli delle variabili semplici. (Vedi il paragrafo Matrici per maggiori informazioni). In uno stesso programma non si può usare lo stesso nome per una variabile semplice e per una matrice, nè per due matrici con diverse dimensioni. Per esempio, non si possono usare Z e Z(3) come nomi di variabili nello stesso programma, e nemmeno X(3,4) e X(2,1,3). Si può invece usare un nome per una variabile numerica e lo stesso seguito dal dollaro per una variabile di stringa (X e X$).

Nomi di Variabili Numeriche

Validi - X, A9, ALPHA, BASE-PAY, V(3), T(X,3),
TABLE(X, XX7Y/2)

Non validi - X$, X/8, 3Y

Nomi di variabili di stringa

Validi - S$, YZ2$, NAME$, Q5$(3,X)
Non validi - S$3, X9, 4Z$

Se inserite un nome di variabile con più di 15 caratteri, compare il messaggio "BAD NAME" e la linea non viene memorizzata.

Le parole riservate non possono essere usate come nomi di variabili, ma come parte di un nome di variabile. Ad esempio, non si può usare LIST ma LIST$.

In ogni istante durante l'esecuzione del programma, ogni variabile ha un unico valore. All'inizio dell'esecuzione ogni variabile numerica è posta = zero, e ogni variabile di stringa ha lunghezza zero. Durante l'esecuzione, viene assegnato un valore alle variabili con un'istruzione LET, READ, FOR - TO - STEP o INPUT. La lunghezza di una variabile di stringa può variare da 0 a 255.

Esempi:

```
>110 ABCDEFGJKLMNOPQ=3
 * BAD NAME
```

Parole Riservate

Le parole riservate non possono essere usate come nomi di variabili in TI BASIC, mentre si possono usare come parte di un nome di variabile (ALEN e LENGTH sono consentite). Segue un elenco completo di tutte le parole riservate di TI BASIC:

ABS
APPEND
ASC
ATN
BASE
BREAK
BYE
CALL
CHR$
CLOSE
CON
CONTINUE
COS
DATA
DEF
DELETE
DIM
DISPLAY
EDIT
ELSE
END
EOF
EXP
FIXED
FOR
GO
GOSUB
GOTO
IF
INPUT
INT
INTERNAL
LEN
LET
LIST
LOG
NEW
NEXT
NUM
NUMBER
OLD
ON
OPEN
OPTION
OUTPUT
PERMANENT
POS
PRINT
RANDOMIZE
READ
REC
RELATIVE
REM
RES
RESEQUENCE
RESTORE
RETURN
RND
RUN
SAVE
SEG $
SEQUENTIAL
SGN
SIN
SQR
STEP
STOP
STR$
SUB
TAB
TAN
THEN
TO
TRACE
UNBREAK
UNTRACE
UPDATE
VAL
VARIABILE

Espressioni Numeriche

Le espressioni numeriche sono costituite da variabili numeriche, costanti numeriche e riferimenti a funzioni, separati da operatori aritmetici (+ - * / ∧). La funzioni richiamate in un'espressione possono essere funzioni di TI BASIC (vedi il paragrafo Funzioni aritmetiche) o funzioni definite dall'utente con un'istruzione DEF. Ci sono due tipi di operatori aritmetici, *prefissi* e *infissi*.

Gli operatori *prefissi* sono il più (+) e il meno (-), e si usano per indicare il segno (positivo o negativo) delle costanti e delle variabili. Il segno più significa che il numero seguente deve essere moltiplicato per +1, e il meno che deve essere moltiplicato per -1. In mancanza di prefisso, il numero viene considerato positivo. Vediamo alcuni esempi di operatori prefissi con costanti e variabili:

10 — 6 + 3
+ A — W

Gli operatori *infissi* servono per i calcoli e comprendono la addizione (+), la sottrazione (-), la moltiplicazione (*), la divisione (/) e l'elevamento a potenza (∧). Ci deve essere un operatore infisso tra ogni costante e/o variabile in un'espressione numerica. Notate che le moltiplicazioni non possono essere espresse in modo implicito mettendo una variabile di fianco all'altra, o con delle parentesi; dovete usare l'operatore di moltiplicazione (*).

Esempi:

```
>NEW

>100 A=6
>110 B=4
>120 C=20
>130 D=2
>140 PRINT A*B/2
>150 PRINT C-D*3+6
>160 END
>RUN
  12
  20

 ** DONE **
```

Nel calcolo delle espressioni aritmetiche, TI BASIC usa le priorità standard.

1. Le espressioni tra parentesi sono calcolate per prime.
2. Poi vengono eseguite gli elevamenti a potenza nell'ordine da sinistra verso destra.
3. Poi vengono eseguite le operazioni con operatori prefissi.
4. Poi vengono eseguite moltiplicazioni e divisioni.
5. Infine si eseguono le addizioni e le sottrazioni.

```
>NEW

>100 A=2
>110 B=3
>120 C=4
>130 PRINT A*(B+2)
>140 PRINT B^A-4
>150 PRINT -C^A;(-C)^A
>160 PRINT 10-B*C/6
>170 END
>RUN
  10
  5
 -16  16
  8

 ** DONE **
```

Notate che 0 ∧ 0 è definito uguale a 1.

```
>PRINT 0^0
  1
```

Espressioni relazionali

Sono espressioni generalmente usate nell'istruzione IF - THEN - ELSE, ma possono essere usate dovunque sono lecite le espressioni numeriche. Quando usate espressioni relazionali all'interno di espressioni numeriche, assumono il valore -1 se la relazione è vera, e il valore 0 se è falsa.

Le operazioni relazionali sono eseguite da sinistra verso destra *prima* della concatenazione delle stringhe e *dopo* tutte le operazioni aritmetiche. Per eseguire concatenazioni di stringhe prima delle operazioni relazionali e/o operazioni relazionali prima delle operazioni aritmetiche, dovete usare le parentesi. Gli operatori relazionali sono

- Uguale a (=)
- Diverso da (<>)
- Minore di (<)
- Minore o uguale (< =)
- Maggiore di (>)
- Maggiore o uguale (>=)

Nella spiegazione dell'istruzione IF - THEN - ELSE vedrete come vengono eseguiti i confronti tra stringhe per assegnare un risultato vero o falso. Ricordate che il risultato che si ottiene dal calcolo di un'operazione relazionale è sempre un numero. Se cercate di usare il risultato come stringa, avrete un errore.

Esempi:

```
>NEW

>100 A=2<3
>110 B=3<=2
>120 PRINT A;B
>130 END
>RUN
 -1  0

 ** DONE **

>NEW

>100 A$="AB"
>110 B$="CDEF"
>120 PRINT (A$&B$)="AB"
>130 END
>RUN
  0

 ** DONE **

>120 PRINT (A$&B$)>"AB"
>130 END
>RUN
 -1

 ** DONE **

>120 PRINT (A$>B$)-4
>RUN
 -4

 ** DONE **

>NEW

>100 A=2<4+3
>110 B=A=0
>120 PRINT A;B
>130 END
>RUN
 -1  0

 ** DONE **
```

Comandi

Introduzione

Esempi:

Quando in alto sul video compaiono il prompt e il cursore intermittente (> ■), il calcolatore è in Modo Comando (o Immediato). In questa situazione, potete dare uno dei comandi illustrati in questo paragrafo. I comandi possono essere inseriti senza numero di linea, e vengono eseguiti immediatamente. Anche molte delle istruzioni TI BASIC possono essere inserite come comandi.

Alcuni dei comandi che vedremo possono essere usati come istruzioni; in questo caso lo segnaleremo nella relativa descrizione.

NEW

NEW

Questo comando cancella il programma che si trova in memoria, e annulla l'effetto dei comandi BREAK e TRACE; inoltre chiude tutti i file aperti (vedi l'istruzione OPEN) e libera lo spazio riservato per caratteri speciali. Ancora il comando NEW cancella tutti i valori delle variabili e la tabella in cui sono registrati i nomi delle variabili. Dopo che è stato eseguito, il video viene cancellato e compare il messaggio "TI BASIC READY". Il prompt e il cursore (■) stanno a indicare che potete inserire un nuovo comando o una linea di programma.

```
 TI BASIC READY
>■
```

LIST

LIST {*[lista linee]* | *"nome dispositivo"* [*: lista linee*]}

Quando viene dato il comando LIST, vengono visualizzate sul video le linee specificate nella *lista linee*. Se date un *nome dispositivo*, le linee vengono stampate sul dispositivo indicato. I *nomi dispositivi* sono dati nei manuali di ogni dispositivo e vanno scritti in lettere maiuscole. In assenza del *nome dispositivo*, le linee sono visualizzate sul video.

Se il comando LIST viene dato senza *lista linee*, viene listato tutto il programma, con il numero di linee in ordine crescente.

```
>NEW
>100 A=279.3
>120 PRINT A;B
>110 B=-456.8
>130 END
>LIST
 100 A=279.3
 110 B=-456.8
 120 PRINT A;B
 130 END
```

LIST

La *lista linee* può essere un numero da solo, un numero preceduto da trattino (per esempio -10), un numero seguito da un trattino (per esempio 10-), o una coppia di numeri separati da trattino. Se è

Esempi:

- Un numero da solo, viene visualizzata la linea corrispondente.

```
>LIST 110
 110 B=-456.8
```

- Un numero preceduto da trattino, vengono listate tutte le linee con numeri di linea minori o uguali a quello specificato.

```
>LIST -110
 100 A=279.3
 110 B=-456.8
```

- Un numero seguito da trattino, vengono listate tutte le linee con numero di linea maggiore o uguale a quello specificato.

```
>LIST 120-
 120 PRINT A;B
 130 END
```

- Una coppia di numeri separati da trattino, vengono listate tutte le linee con numero di linea maggiore o uguale al primo, e minore o uguale del secondo.

```
>LIST 90-120
 100 A=279.3
 110 B=-456.8
 120 PRINT A;B
```

Se nel programma in memoria non ci sono linee con numeri corrispondenti a quelli indicati nella lista linee, si seguono le regole seguenti

- Se i numeri di linea delle liste sono maggiori di quelli del programma, viene visualizzata la linea con il numero maggiore.

```
>LIST 150-
 130 END
```

- Se i numeri di linea delle liste sono minori di quelli del programma, viene visualizzata la linea con il numero minore.

```
>LIST -90
 100 A=279.3
```

- Se i numeri di linea sono compresi tra quelli del programma, viene visualizzata la linea con il numero immediatamente superiore.

```
>LIST 105
 110 B=-456.8
```

- Se il numero di linea è 0 o maggiore di 32767, compare il massaggio BAD LINE NUMBER

```
>LIST 0
 * BAD LINE NUMBER

>LIST 33961
 * BAD LINE NUMBER
```

- Se il numero di linea non è intero, compare il messaggio INCORRECT STATEMENT.

```
>LIST 32.7
 * INCORRECT STATEMENT
```

- Se non è presente un programma in memoria, compare il messaggio CAN'T DO THAT.

```
>NEW

>LIST
 * CAN'T DO THAT
```

- Per interrompere la lista premete il tasto **FCTN 4 (CLEAR)**.
- Comando LIST per dispositivi accessori.
  LIST può essere usato per produrre la lista su un dispositivo particolare. Per esempio, LIST"TP" fa sì che il programma venga stampato, se al calcolatore è collegata una stampante termica SOLID STATE TI.

Notate che il nome del dispositivo deve essere tra virgolette e in caratteri maiuscoli. Per ulteriori informazioni vi rimandiamo ai manuali dei singoli dispositivi.

Comandi	**Linee visualizzate**
LIST	Tutte le linee del programma
LIST x	La linea con numero x
LIST x-y	Le linee tra x e y, compres i
LIST x-	Le linee maggiori o uguali a x
LIST -y	Le linee minori o uguali a y

RUN

RUN [*numero linea*]

Il comando RUN manda in esecuzione il programma che si trova in memoria. Prima di iniziare l'esecuzione,tutte le variabili numeriche sono messe a zero, la lunghezza di tutte le variabili di stringa è messa a zero e viene liberato tutto lo spazio precedentemente riservato a caratteri grafici speciali.

Se non è specificato il numero di linea, l'esecuzione inizia dalla linea con il numero di linea più basso.

Se è specificato il numero di linea, l'esecuzione inizia da quella linea. Notate in questo esempio che, poichè l'esecuzione inizia dalla linea 110, il valore di A resta a zero.

Se specificate un numero di linea che non è nel programma, compare il messaggio "BAD LINE NUMBER"

Se date il comando RUN quando in memoria non c'è alcun programma, compare il messaggio "CAN'T DO THAT"

***In** Extended Basic il comando RUN può essere usato come istruzione.*

Esempi:

```
>NEW
>100 A=-16
>110 B=25
>120 PRINT A;B
>130 END
>RUN
 -16  25

 ** DONE **

>RUN 110
  0  25

 ** DONE **

>RUN 115
 * BAD LINE NUMBER

>NEW
>RUN

* CAN'T DO THAT
```

BYE

BYE

Quando avete finito di lavorare e volete uscire dal BASIC, date il comando BYE. Vi raccomandiamo di usare sempre il comando BYE (e non QUIT) per uscire dal BASIC. Quando date il BYE, la prima cosa che fa il calcolatore è di chiudere tutti i file aperti (vedi l'istruzione OPEN). Poi vengono cancellati il programma e i valori di tutte le variabili. Infine il calcolatore è rimesso in condizione di ripartire con il BASIC, quando voi lo volete. Dopo che il comado BYE è eseguito , sul video ricompare il Titolo Principale.

Esempi:

```
>NEW

>100 LET X$="CIAO, GENIO"
>110 PRINT X$
>120 END
>RUN
 CIAO, GENIO

 ** DONE **

>BYE
```

NUMBER

{NUMBER / NUM} [*linea iniziale*] [,*incremento*]

Con il comando NUMBER il vostro calcolatore genera automaticamente i numeri di linea, e si dice che è in Modo "Number".

Il primo numero è quello specificato come *linea iniziale*. Il secondo numero dopo la virgola definisce l'incremento per generare i numeri successivi. Per uscire dal Modo "Number", premete **ENTER** dopo che viene visualizzato il numero generato. La linea vuota non viene aggiunta al programma.

Se non sono specificati la *linea iniziale* e l'*incremento* si assume 100 come *linea iniziale* e 10 come *incremento*.

Se specificate solo la *linea iniziale*, si assume 10 come incremento.

Se specificate solo l'*incremento*, si assume 100 come *linea iniziale*. Notate la virgola prima del cinque nell'esempio.

Esempi:

>NEW

>NEW

>NEW

>NEW

NUMBER

Quando siete in Modo "Number", se il numero di linea generato corrisponde a una linea già presente nel programma, viene visualizzata questa linea. Notate che quando viene visualizzata una linea di programma in Modo "Number", non compare il prompt (>) a sinistra del numero. Questo vi segnala che la linea esiste già nel programma, e potete modificarla. Per maggiori informazioni sulle modalità di correzione e modifica dei programmi , vedi il paragrafo successivo. Se non volete modificarla, premete **ENTER**, e verrà generato il numero di linea successivo.
In Modo "Number", se inserite una linea di programma e fate un errore, compare un opportuno messaggio di errore e resta visualizzato lo stesso numero di linea. Ribattete la linea correttamente e inseritela nuovamente. Se viene generato un numero di linea maggiore di 32767, il calcolatore esce dal Modo "Number".

Esempi:

Editing in Modo Number

Quando inserite nuove linee o ne modificate di esistenti in Modo "Number". potete usare tutti i tasti speciali di correzione. Alcuni di questi funzionano in modo diverso che in Modo "Comandi".

ENTER - Questo tasto ha diverse funzioni a seconda della situazione in cui viene usato. Le vediamo di seguito.

- Se premete **ENTER** subito dopo che è stato generato un numero di linea, il calcolatore esce dal Modo "Number".
- Se inserite un'istruzione dopo il numero generato e premete **ENTER**, la nuova linea è aggiunta al programma, e viene generato il numero successivo.
- Se è visualizzata una linea di un programma e subito dopo premete **ENTER**, la linea resta così com'è nel programma, e viene generato il numero successivo.
- Se è visualizzata una linea di un programma e voi cancellate il testo (lasciando solo il numero) e premete **ENTER**, il calcolatore esce dal Modo "Number", e la linea non è eliminata dal programma.
- Se modificate una linea di un programma, dopo che è stata visualizzata e premete **ENTER**, la linea del programma viene sostituita con quella modificata, e viene generato il numero successivo.

FCTN 4 (CLEAR) - Se premete il tasto CLEAR in qualsiasi momento in Modo "Number", la linea scorre sul video, e il calcolatore esce dal Modo "Number". Tutte le modifiche fatte sulla linea prima di premere CLEAR sono ignorate. Perciò, se state modificando una linea di un programma, questa non verrà modificata. Se invece state inserendo una nuova linea , questa non verrà aggiunta al programma.

RESEQUENCE

{RESEQUENCE / RES} [*linea iniziale*] [, *incremento*]

Se volete cambiare i numeri delle linee di un programma, date il comando RESEQUENCE. In base alla linea iniziale e all'incremento specificati, alle linee del programma verranno assegnati nuovi numeri. Il comando RES si dà esattamente come il NUMBER.

Nota - Se in un'istruzione del programma c'è un riferimento ad un numero di linea inesistente, il riferimento viene cambiato in 32767, e non viene inviato alcun messaggio.

Esempi:

```
>NEW

>100 A=27.9
>110 B=34.1
>120 PRINT A;B
>130 END

>RESEQUENCE 20,5
>LIST
 20 A=27.9
 25 B=34.1
 30 PRINT A;B
 35 END
```

```
>NEW

>100 Z=Z+2
>110 PRINT Z
>120 IF Z=50 THEN 150
>130 GO TO 100
>140 END
>RES 10,5
>LIST
 10 Z=Z+2
 15 PRINT Z
 20 IF Z=50 THEN 32767
 25 GO TO 10
 30 END
```

BREAK

BREAK *lista linee*

Con il comando BREAK si inseriscono nel programma dei punti di arresto che possono servire per trovare eventuali errori. Quando inserite un punto di arresto con il comando BREAK, dite al calcolatore di interrompere l'esecuzione del programma prima di eseguire l'istruzione che si trova in quel punto.

La *lista linee* è una lista di numeri delle linee in cui volete inserire i punti di arresto. I numeri sono separati da virgole (per esempio BREAK 10,23,35). Naturalmente può esserci un solo numero.

Ogni volta che durante l'esecuzione il programma raggiunge una linea con un punto di arresto, si ferma prima di eseguire l'istruzione corrispondente, compare il messaggio "BREAKPOINT AT numero di linea" e il cursore intermittente vi chiede di dare un comando.

Esempi:

```
>NEW

>100 A=26.7
>110 C=19.3
>120 PRINT A
>130 PRINT C
>140 END

>BREAK 110

>RUN
  * BREAKPOINT AT 110
>■
```

BREAK

Quando il programma si ferma su di un punto di arresto, potete dare un qualsiasi comando, o un'istruzione che può essere usata come comando. I valori delle variabili non subiscono alcun cambiamento, a meno che inseriate un'istruzione che le modifichi. Notate che nel nostro esempio C è ancora uguale a zero perchè l'istruzione 110 non è ancora stata eseguita.

Esempi:

```
>NEW

>100 B=29.7
>110 BREAK 120,140
>120 H=15.8
>130 PRINT B
>140 PRINT H
>150 END
>RUN

 * BREAKPOINT AT 120
```

Il comando BREAK può essere usato anche come istruzione. In questo caso, se c'è la *lista linee*, i punti di arresto sono messi alle linee specificate, e possono essere eliminati come abbiamo visto, e (ricordatevelo) vengono messi ogni volta che l'istruzione è eseguita.

```
>110 BREAK
>RUN

 * BREAKPOINT AT 110
```

Se il comando BREAK è usato come istruzione senza *lista linee*, l'istruzione stessa agisce da punto di arresto, e ogni volta che è eseguita, il programma si ferma. L'unico modo per evitarlo è di cancellare la linea dal programma. Notate che un comando BREAK senza *lista linee* può essere inserito solo come istruzione di programma.

```
>LIST 110
 110 C=19.3
>PRINT A;C
 26.7  0
```

Se nella *lista linee* c'è un numero di linea che non esiste nel programma, compare il messaggio "BAD LINE NUMBER", e i punti di arresto sono inseriti alle linee specificate che esistono nel programma.

```
>110 BREAK 125,140
>RUN

 * WARNING:
   BAD LINE NUMBER IN 110
```

CONTINUE

{CONTINUE
CON}

Il comando CONTINUE può essere dato per far ripartire un programma fermo su di un punto di arresto. Per i punti di arresto e come inserirli, vedi il comando BREAK. Ricordate che il programma può essere interrotto anche premendo il tasto **CLEAR** durante la sua esecuzione.

Non potete dare il comando CONTINUE se, durante l'arresto del programma, l'avete modificato (aggiunto, cancellato o cambiato linee). Questo per evitare gli errori che potrebbero capitare rilanciando da un punto intermedio un programma che è stato modificato. Se cercate di farlo, sul video compare il messaggio "CAN'T CONTINUE"

Esempi:

```
>NEW

>100 A=9.6
>110 PRINT A
>120 END
>BREAK 110

>RUN

 * BREAKPOINT AT 110
>CONTINUE

> 9.6


 ** DONE **
>BREAK 110

>RUN

 * BREAKPOINT AT 110

>100 A=10.1

>CONTINUE
 * CAN'T CONTINUE
```

UNBREAK

UNBREAK [*lista linee*]

Serve per eliminare i punti di arresto dalle linee specificate nella *lista linee*. Per una spiegazione sui punti di arresto, vedi il comando BREAK.

La *lista linee* è una lista di numeri delle linee da cui volete eliminare i punti di arresto. I numeri sono separati da virgole (per esempio UNBREAK 10,23). Se c'è un solo numero non sono necessarie le virgole.

Esempi:

```
>NEW

>100 A=26.7
>110 C=19.3
>120 PRINT A
>130 PRINT C
>140 END
>BREAK 110,130

>RUN

 * BREAKPOINT AT 110


>UNBREAK 130

>CONTINUE
  26.7
  19.3

 ** DONE **
```

UNBREAK

Se date un comando UNBREAK senza *lista linee*, vengono eliminati tutti i punti di arresto definiti con un comando o un'istruzione di BREAK precedenti. Notate che il comando UNBREAK non ha effetto su un'istruzione BREAK senza *lista linee*. L'unico modo per impedire che il programma si fermi a un'istruzione BREAK senza *lista linee* è di cancellare l'istruzione stessa.

Esempi:

```
>125 BREAK
>BREAK 100,120,130

>RUN

 * BREAKPOINT AT 100
>UNBREAK

>CONTINUE
  26.7

 * BREAKPOINT AT 125

>CONTINUE
  19.3

 ** DONE **
```

Il comando UNBREAK può anche essere usato come istruzione in un programma, e ha lo stesso effetto. Notate nell'esempio che l'istruzione UNBREAK elimina il punto di arresto alla linea 130.

```
>BREAK 130

>125 UNBREAK 130
>RUN
  26.7
  19.3

 ** DONE **
```

Se nella *lista linee* c'è un numero di linea valido, ma inesistente nel programma, compare il messaggio "BAD LINE NUMBER", e vengono eliminati i punti di arresto dalle linee specificate esistenti.

```
>BREAK 130

>UNBREAK 130,105
 * WARNING:
   BAD LINE NUMBER

>RUN
  26.7
  19.3

 ** DONE **
```

TRACE

TRACE

Il comando TRACE vi consente di vedere l'ordine in cui il calcolatore esegue le istruzioni di un programma. Quando lo date, viene visualizzato il numero di ogni linea prima che sia eseguita. E' molto utile per trovare errori come cicli senza fine.

Esempi:

```
>NEW
>100 PRINT "ORA"
>110 B=27.9
>120 PRINT :B
>130 END
>TRACE

>RUN
 <100>ORA
  <110><120>
   27.9
  <130>
  ** DONE **
```

UNTRACE

UNTRACE

Il comando UNTRACE annulla gli effetti del TRACE, e può essere usato come istruzione di programma.

Esempi:

```
>NEW
>100 FOR I=1 TO 2
>110 PRINT I
>120 NEXT I
>130 END
>TRACE

>RUN
  <100><110> 1
   <120><110> 2
   <120><130>
  ** DONE **

  >UNTRACE

  >RUN
    1
    2
  ** DONE **
```

EDIT

EDIT *numero di linea*

Potete modificare i programmi entrando in Modo "Edit", cioè dando il comando EDIT seguito da un *numero di linea*. Se specificate un numero di linea che non esiste nel programma, compare il messaggio "BAD LINE NUMBER".

Quando entrate in Modo "Edit", la linea richiesta compare sullo schermo senza il carattere di controllo (>). Potete modificare qualunque carattere della linea tranne il numero di linea. I tasti speciali per modifiche e correzioni con la loro funzione in Modo "Edit" sono presentati di seguito.

ENTER - Quando premete **ENTER**, tutte le modifiche effettuate diventano definitive, e il calcolatore esce dal Modo "Edit". Se avete cancellato tutto il testo della linea e premete **ENTER**, la linea viene cancellata.

FCTN ↑ (UP) - Quando premete questo tasto, le modifiche effettuate alla linea diventano definitive, e viene visualizzata la linea con il numero immediatamente più basso. Se non ce n'è, il calcolatore esce dal Modo "Edit".

FCTN ↓ (DOWN) - Quando premete questo tasto, tutte le modifiche effettuate diventano definitive, e viene visualizzata la linea con il numero immediatamente più alto. Se non ce n'è, il calcolatore esce dal Modo "Edit".

SAVE

SAVE *nome file*

Il comando SAVE vi consente di copiare il vostro programma dalla memoria su un altro dispositivo. Con il comando OLD potete in seguito riportarlo in memoria per eseguirlo o modificarlo.

Vi diamo una breve spiegazione prendendo un registratore a cassetta come dispositivo di memoria.

Selezionate il registratore che volete usare, dando come *nome file* CS1 o CS2 dopo il comando SAVE. Dopo aver collegato il registratore, date il comando SAVE e premete **ENTER**. Il calcolatore invia sul video le istruzioni per aiutarvi a capire la procedura di SAVE.

SAVE

Nella colonna a destra ci sono le istruzioni generate dal calcolatore. Usiamo CS1 come esempio, ma è lo stesso con CS2.

Quando date il comando SAVE, il calcolatore vi dice come usare il registratore, come si vede a destra. Dopo che il programma è stato registrato, vi chiede se volete controllare il nastro per assicurarvi che il programma sia stato copiato correttamente. Se rispondete **N**, ricompare il cursore intermittente a sinistra sul video, e potete dare un altro comando qualsiasi. Se premete **Y**, appaiono le istruzioni per attivare il registratore.

Nota - Le singole lettere di risposta (**Y,N,R**,etc.) che date durante la procedura SAVE devono essere maiuscole. Tenete premuto il tasto **SHIFT** e premete la lettera opportuna.

Se capita un errore, potete scegliere trà una delle tre possibilità seguenti.

- Premete **R** per registrare nuovamente il programma. Compariranno le stesse istruzioni di prima per guidarvi.
- Premete **C** per ripetere le procedure di controllo. Può darsi che dobbiate regolare il volume o il tono del registratore.
- Premete **E** per "uscire" dalla procedura di registrazione. Il calcolatore vi dirà di fermare il registratore e premere **ENTER**. Vedrete un messaggio di errore sullo schermo. Significa che la routine SAVE non ha registrato correttamente il vostro programma. Dopo aver controllato il vostro registratore, potete provare a registrare nuovamente il programma. Quando ricompare sul video il cursore intermittente, date il comando BASIC che volete.

Quando il comando SAVE viene eseguito il programma resta comunque in memoria.

(Per una spiegazione più dettagliata vedi il paragrafo "Connettore per interfaccia col registratore a cassetta"). Le istruzioni per usare il Sistema di Memoria a Dischi TI sono nel manuale della "Scheda di Controllo per Unità a dischi".

Con il comando MERGE dell'Extended BASIC, possono essere fusi due programmi.

Esempi:

```
>SAVE CS1

 * REWIND CASSETTE TAPE      CS1
   THEN PRESS ENTER

 * PRESS CASSETTE RECORD     CS1
   THEN PRESS ENTER

 * RECORDING

 * PRESS CASSETTE STOP       CS1
   THEN PRESS ENTER

 * CHECK TAPE (Y OR N)

 * REWIND CASSETTE TAPE      CS1
   THEN PRESS ENTER

 * PRESS CASSETTE PLAY       CS1
   THEN PRESS ENTER

 * CHECKING

 * DATA OK

 * PRESS CASSETTE STOP       CS1
   THEN PRESS ENTER


 * ERROR - NO DATA FOUND
   PRESS R TO RECORD
   PRESS C TO CHECK
   PRESS E TO EXIT


 * ERROR IN DATA DETECT
   PRESS R TO RECORD
   PRESS C TO CHECK
   PRESS E TO EXIT


 * I/O ERROR 66
```

OLD

OLD *nome file*

Il comando OLD riporta in memoria un programma precedentemente registrato su di un altro dispositivo con la SAVE. A questo punto potete eseguirlo, listarlo o modificarlo. Vi diamo qui le spiegazioni per usare il registratore (CS1) con il comando OLD. Le istruzioni per il Sistema di Memoria a Dischi TI sono nel manuale della "Scheda di Controllo per Unità a Dischi".

Dopo aver dato il comando OLD e premuto **ENTER**, il calcolatore vi dà sul video le istruzioni per aiutarvi nelle operazioni. Assicuratevi di aver collegato il registratore e inserito la cassetta giusta.

Nella colonna di destra riportiamo le istruzioni che compaiono sullo schermo quando date il comando OLD. Troverete una descrizione dettagliata di queste procedure nel paragrafo "Il registratore": memoria esterna".

Se il calcolatore non ha caricato correttamente il vostro programma in memoria, si avrà un errore, e avete una delle seguenti possibilità

- Premete **R** per ripetere la procedura di lettura. Prima però eseguite i controlli prescritti nel paragrafo sul registratore.
- Premete **E** per uscire dalla procedura di lettura. Comparirà sullo schermo un messaggio di errore per segnalarvi che il calcolatore non ha letto correttamente il programma in memoria.

Nota - Le risposte (**E o R**) che date durante la routine OLD devono essere maiuscole. Tenete premuto il tasto **SHIFT** e premete la lettera opportuna.

Quando il cursore intermittente ricompare sul video, potete dare qualsiasi comando BASIC.

Anche se il programma non è stato letto correttamente nella memoria, può risultare cancellata una parte di quello già presente in memoria. Usate il comando LIST per controllare il contenuto della memoria prima di procedere.

Esempi:

```
>OLD CS1

* REWIND CASSETTE TAPE CS1
  THEN PRESS ENTER

* PRESS CASSETTE PLAY  CS1
  THEN PRESS ENTER

* READING

* DATA OK

* PRESS CASSETTE STOP  CS1
  THEN PRESS ENTER

* ERROR - NO DATA FOUND
  PRESS R TO READ
  PRESS E TO EXIT

* I/O ERROR 56
```

DELETE

DELETE $\begin{Bmatrix} \textit{nome file} \\ \textit{nome programma} \end{Bmatrix}$

Il comando DELETE vi serve per cancellare un programma o un file da un disco. Il *nome file* e il *nome programma* sono espressioni di stringa. Se usate una costante di stringa, dovete metterela tra virgolette.

Poichè questi comandi sono usati soprattutto con il sistema a floppy disk, vi consigliamo di leggere il manuale relativo per avere più informazioni.

Esempi:

```
>SAVE NOME$

>DELETE NOME$
>DELETE "CS1"
```

Istruzioni di Assegnazione e Controllo

Introduzione

Questo paragrafo descrive le istruzioni di un programma che non eseguono funzioni di ingresso/uscita. Sono la LET, che assegna valori alle variabili, la STOP, la END e la REM, e le istruzioni che controllano la strada che il calcolatore percorre nell'eseguire il programma. Queste istruzioni di controllo, GO TO, ON GO TO, IF-THEN-ELSE, FOR-TO-STEP, NEXT vi permettono di creare facilmente dei cicli e dei salti condizionati o no. Con le istruzioni di questo paragrafo e quelle di ingresso/uscita potrete scrivere programmi divertenti e utili.

In TI BASIC potete scrivere solo un'istruzione per linea, mentre con il TI EXTENDED BASIC potete scrivere più di un'istruzione per linea. Questo riduce il tempo di esecuzione di un programma, risparmia memoria e consente di programmare un'unità logica (come un ciclo FOR NEXT)in una sola linea.

LET (Istruzione di Assegnazione)

[LET] variabile = espressione

Serve per assegnare valori alle variabili del vostro programma. Il calcolatore calcola l'*espressione* a destra dell'uguale e assegna il risultato alla *variabile* che compare alla sinistra.

La *variabile* e l'*espressione* devono essere dello stesso tipo. Espressioni numeriche devono essere assegnate a variabili numeriche, espressioni di stringa a variabili di stringa. Vengono seguite le solite regole in caso di overflow o di underflow nel calcolo di un'espressione numerica. Rivedete il paragrafo "Costanti numeriche" per ulteriori informazioni. Se la stringa risultante da un'espressione di stringa ha più di 255 caratteri, viene troncata sulla destra e il programma continua. Non viene inviato alcun messaggio.

Potete usare operatori relazionali in espressioni numeriche o di stringa. Il risultato è -1 se la relazione è vera, 0 se è falsa. Vedi il paragrafo "Espressioni Relazionali" per ulteriori informazioni.

Esempi:

```
>NEW

>100 LET M=1000
>110 LET C=186000
>120 E=M*C^2
>130 PRINT E
>140 END
>RUN
  3.4596E+13

 ** DONE **
```

```
>NEW

>100 LET X$="CIAO, "
>110 NOME$="GENIO"
>120 PRINT X$;NOME$
>130 END
>RUN
 CIAO, GENIO

 ** DONE **
```

```
>NEW

>100 LET A=20
>110 B=10
>120 LET C=A>B
>130 PRINT A;B;C
>140 C=A<B
>150 PRINT A;B;C
>160 END
>RUN
  20  10 -1
  20  10  0

 ** DONE **
```

REM

REM commento

L'istruzione REM vi consente di documentare il vostro programma, inserendo commenti nel programma stesso. Quando il calcolatore incontra una REM durante l'esecuzione di un programma, non fa niente. La REM serve solo come documentazione e spiegazione per l'utente.

Esempi:

```
>NEW

>100 REM NUMERI DA 1 A 10

>110 FOR X=1 TO 10
>120 PRINT X;
>130 NEXT X
>140 END
>RUN
 1  2  3  4  5  6  7  8  9
 10
 ** DONE **
```

In un'istruzione REM potete usare qualsiasi carattere stampabile. La sua lunghezza è quella di una qualsiasi linea di programma (112 caratteri o quattro linee di video).

```
>NEW

>100 A=762
>110 B=425
>120 REM STAMPA A+B
>130 PRINT A+B
>140 END
>RUN
  1187

 ** DONE **
```

END

END

L'istruzione END mette fine all'esecuzione del vostro programma, e può essere usata al posto della STOP in TI BASIC. Può apparire in qualsiasi punto del programma, ma normalmente è nella linea con il numero più alto, in modo da rappresentare la fine fisica e logica del programma, mentre si usa la STOP per altri punti di arresto intermedi. In TI BASIC non è indispensabile l'istruzione END.

Esempi:

```
>NEW

>100 A=10
>110 B=20
>120 C=A*B
>130 PRINT C
>140 END
>RUN
  200
```

STOP

STOP

L'istruzione STOP mette fine all'esecuzione di un programma quando viene eseguita,e può essere usata al posto della END in TI BASIC. Può essere messa in qualsiasi punto del programma. e ce ne possono essere diverse in uno stesso programma.

Esempi:

```
>NEW
>100 CALL CLEAR
>110 FOR I=1 TO 15
>120 CALL HCHAR(1,1,42,768)
>130 GOSUB 160
>140 NEXT I
>150 STOP
>160 F=I
>170 B=I+1
>180 CALL COLOR(2,F,B)
>190 RETURN
>200 END
>RUN
  --LO SCHERMO SI RIEMPIE
    DI ASTERISCHI E CAMBIA
    COLORE 15 VOLTE
```

GO TO

{GOTO / GO TO} *numero di linea*

L'istruzione GO TO serve per trasferire il controllo in avanti o indietro in un programma. Quando il calcolatore incontra un'istruzione GOTO, salta sempre all'istruzione indicata dal *numero di linea.* Si chiama salto *incondizionato.*

Nel programma a destra, la linea 170 è un salto *incondizionato.* Quando arriva a questo punto il calcolatore salta sempre alla linea 140. Se il *numero di linea* non esiste nel programma, quando incontra l'istruzione il calcolatore si ferma e invia il messaggio "BAD LINE NUMBER".

Notate che lo spazio tra GO e TO è opzionale.

Esempi:

```
>NEW

>100 CALL CLEAR
>110 REM ESEMPIO DI GOTO
>140 PRINT B^A-4
>150 PRINT "INSTRUMENTS"
>160 PRINT " "
>170 GOTO 140
>180 END
>RUN
```

ON-GOTO

ON *espressione numerica* {GOTO / GO TO} *numero di linea* [,*numero di linea*] ...

L'istruzione ON-GOTO dice al calcolatore di saltare a una tra diverse linee di programma, a seconda del valore dell'*espressione numerica.*

Il calcolatore dapprima calcola il risultato dell'*espressione numerica* e lo arrotonda a un intero. Questo numero diventa un puntatore, che dice al calcolatore quale delle linee specificate nella ON-GOTO deve essere eseguita. Se il valore calcolato è 1, il calcolatore salta all'istruzione corrispondente al primo numero di linea specificato. Se il valore è 2, salta all'istruzione corrispondente al secondo numero , e così via.

Se il valore dell'*espressione numerica*, arrotondato, è minore di 1 o maggiore del numero di *numeri di linea* specificati, il programma si arresta e compare il messaggio "BAD VALUE IN xx". Se i *numeri di linea* specificati sono diversi da quelli presenti nel programma, compare il messaggio "BAD LINE NUMBER" e il programma si arresta.

Esempi:

```
>NEW
>100 REM DOVE ANDIAMO?
>110 INPUT X
>120 ON X GOTO 130,150,170
 ,190,210
>130 PRINT "X=1"
>140 GOTO 110
>150 PRINT "X=2"
>160 GOTO 110
>170 PRINT "X=3"
>180 GOTO 110
>190 PRINT "X=4"
>200 GOTO 110
>210 END
>RUN
 ? 2
 X=2
 ? 1.2
 X=1
 ? 3.7
 X=4
 ? 6

 * BAD VALUE IN 120
```

IF-THEN-ELSE

IF { espressione relazionale / espressione numerica } THEN linea 1 [ELSE linea 2]

L'istruzione IF-THEN-ELSE vi consente di cambiare la normale esecuzione in sequenza di un programma con un salto *condizionato*.

Il calcolatore calcola il valore dell'espressione che compare nell'istruzione, come A > 50. Se l'espressione è vera, salta alla *linea* 1 che segue la parola THEN, altrimenti salta alla *linea 2*, che segue la parola ELSE. Se manca la parola ELSE, continua dalla linea successiva.

Gli operatori relazionali validi in TI BASIC sono

- uguale a (=)
- minore di (<)
- maggiore di (>)
- diverso da (< >)
- minore o uguale a (< =)
- maggiore o uguale a (> =)

Vediamo alcune espressioni relazionali valide.

- A>7
- A$<"YES"
- (A+B)/2< > AVG
- CHR$(L)="A"
- (A$&C$)>=D$

Un'*espressione numerica* deve essere confrontata con un'*espressione numerica*, e un'*espressione di stringa* con un'*espressione di stringa*. Le *espressioni numeriche* vengono confrontate algebricamente; le espressioni di stringa vengono confrontate da sinistra a destra, carattere per carattere, usando il codice ASCII. Un carattere con codice ASCII minore viene considerato minore di un carattere con un codice ASCII maggiore. Così si possono ordinare stringhe in ordine alfabetico o numerico. Se una stringa è più lunga di un'altra, viene eseguito il confronto con ogni carattere della stringa più corta. Se non c'è differenza, viene considerata maggiore la stringa più lunga.

Esempi:

```
>NEW

>100 REM RICERCA MASSIMO
>105 REM TRA N NUMERI
>110 INPUT "QUANTI?":N
>120 INPUT "VALORE?":A
>130 L=A
>140 N=N-1
>150 IF N<=0 THEN 180
>160 INPUT "VALORE?":A
>170 IF L>A THEN 140 ELSE 130

>180 PRINT L;"VAL. MASSIMO"

>190 END
>RUN
 QUANTI?3
 VALORE?456
 VALORE?321
 VALORE?292

  456 VAL. MASSIMO

 ** DONE **
```

```
>NEW

>100 INPUT "A$: ":A$
>110 INPUT "B$: ":B$
>120 IF A$=B$ THEN 160
>130 IF A$<B$ THEN 180
>140 PRINT "B$ MINORE"
>150 GOTO 190
>160 PRINT "A$ = B$"
>170 GOTO 190
>180 PRINT "B$ MAGGIORE"

>190 END
>RUN
 A$: TEXAS
 B$: TEX
 B$ MINORE

 ** DONE **
```

IF-THEN-ELSE

Nell'istruzione IF-THEN-ELSE si può usare in alternativa una *espressione numerica*. Nell'esempio a destra, il calcolatore calcola l'espressione A+B. Se il risultato è zero, l'espressione è considerata falsa, se è diverso da zero è considerata vera. E' lo stesso di

IF *espressione* < > 0 THEN *linea 1*

Esempi:

```
>NEW

>100 INPUT "A: ":A
>110 INPUT "B: ":B
>120 IF A+B THEN 150
>130 PRINT "RISULTATO 0"

>135 PRINT "ESPR. FALSA"

>140 GOTO 100
>150 PRINT "RISULT. NON 0"
>155 PRINT "ESPR. VERA"
>160 GOTO 100
>RUN
 A: 2
 B: 3
 RISULT. NON 0
 ESPR. VERA
 A: 2
 B: -2
 RISULTATO 0
 ESPR. FALSA

(PREMERE CLEAR PER
 USCIRE DAL CICLO)
```

FOR-TO-STEP

FOR *variabile di controllo* = valore iniziale TO *estremo* [STEP *incremento*]

L'istruzione FOR-TO-STEP serve per programmare facilmente procedimenti ripetitivi (iterativi). In coppia con l'istruzione NEXT è usata per costruire un ciclo FOR-NEXT. Se manca la parola STEP, il calcolatore usa un *incremento* di + 1.

La *variabile di controllo* è una variabile numerica che serve come contatore del ciclo. Quando viene eseguita l'istruzione FOR-TO-STEP alla *variabile di controllo* viene assegnato il *valore iniziale*. Il calcolatore esegue tutte le istruzioni finchè incontra l'istruzione NEXT.

Quando viene eseguita la NEXT, la *variabile di controllo* viene incrementata della quantità specificata nella clausola STEP. (Se l'*incremento* è negativo, la *variabile di controllo* viene in realtà ridotta della quantità specificata.) Poi il calcolatore confronta il valore della *variabile di controllo* con il valore dell'*estremo*. Se non lo supera, vengono ripetute le istruzioni seguenti la FOR- TO-STEP fino alla NEXT. Se il nuovo valore della *variabile di controllo* è superiore all'estremo (nel caso che l'*incremento* sia positivo) o minore (se l'*incremento* è negativo), il calcolatore esce dal ciclo, e passa a eseguire le istruzioni successive alla NEXT. il valore della *variabile di controllo* non viene modificato quando il calcolatore esce dal ciclo.

Potete controllare quante volte viene eseguito il ciclo FOR-NEXT con i valori che assegnate nell'istruzione FOR-TO-STEP. L'*estremo* e, se c'è, l'*incremento*, sono espressioni numeriche che vengono calcolate una volta per tutte quando si incontra l'istruzione FOR-TO-STEP, e restano così fino a quando il ciclo è finito. Qualsiasi modifica di questi valori mentre il ciclo è in corso non modifica il numero di volte che questo deve essere ripetuto. Se il valore dell'*incremento* è zero, compare sul video il messaggio "BAD VALUE IN xx" e il programma si ferma.

Esempi:

```
>NEW

>100 REM INTERESSE SEMPL.
>105 REM PER 10 ANNI
>110 INPUT "MONTANTE?":P
>120 INPUT "INTERESSE?":R
>130 FOR ANNI=1 TO 10
>140 P=P+(P*R)
>150 NEXT ANNI
>160 P=INT(P*100+.5)/100
>170 PRINT P
>180 END
>RUN
 MONTANTE? 100
 INTERESSE? .0775
  210.95

 ** DONE **
```

```
>NEW

>100 REM INCR. NON INTERO
>110 FOR X=.1 TO 1 STEP.2
>120 PRINT X;
>130 NEXT X
>140 PRINT :X
>150 END
>RUN

  .1  .3  .5  .7  .9
  1.1

 ** DONE **
```

```
>NEW

>100 L=5
>110 FOR I=1 TO L
>120 L=20
>130 PRINT L;I
>140 NEXT I
>150 END
>RUN
  20  1
  20  2
  20  3
  20  4
  20  5

 ** DONE **
```

FOR-TO-STEP

Quando date un comando RUN, prima che il programma sia eseguito, il calcolatore controlla che ci sia lo stesso numero di FOR-TO-STEP e di NEXT. Se non è così, vi invia il messaggio "FOR-NEXT-ERROR" e il programma non viene eseguito.

Se cambiate il valore della *variabile di controllo* durante l'esecuzione del ciclo, cambiate anche il numero di volte che il ciclo verrà eseguito.

Esempi:

```
>NEW
>100 FOR I=1 TO 10
>110 I=I+1
>120 PRINT I
>130 NEXT I
>140 PRINT I
>150 END
>RUN
  2
  4
  6
  8
  10
  11

 ** DONE **
```

In TI BASIC le espressioni del *valore iniziale*, dell'*estremo* e dell'*incremento* vengono calcolati prima di assegnare il *valore iniziale* alla *variabile di controllo*. Così, nel programma a destra, alla linea 110 il valore 5 viene assegnato all'*estremo* prima di assegnare un valore a I come *variabile di controllo*, e il ciclo viene eseguito 5 volte, non una sola.

```
>NEW
>100 I=5
>110 FOR I=1 TO I
>120 PRINT I;
>130 NEXT I
>140 END
>RUN
  1  2  3  4  5

 ** DONE **
```

Il segno della *variabile di controllo* può cambiare durante l'esecuzione del ciclo.

```
>NEW
>100 FOR I=2 TO -3 STEP -1
>110 PRINT I;
>120 NEXT I
>130 END
>RUN
  2  1  0 -1 -2 -3

 ** DONE **
```

Quando esegue l'istruzione FOR il calcolatore controlla se l'*estremo* supera il *valore iniziale* prima di eseguire il ciclo. Il *valore iniziale non* deve essere necessariamente 1, ma può essere qualsiasi. Se però il *valore iniziale* è maggiore dell'*estremo* e l'*incremento* è positivo, il ciclo non verrà eseguito del tutto, e il calcolatore passa all'istruzione successiva. Così, se l'*incremento* è negativo e il *valore iniziale* è minore dell'*estremo*, il ciclo non verrà eseguito.

```
>NEW
>100 REM VALORE INIZIALE
>105 REM TROPPO GRANDE
>110 FOR I=6 TO 5
>120 PRINT I
>130 NEXT I
>140 END
>RUN

**DONE **
```

FOR-TO-STEP

I cicli FOR-NEXT possono essere "annidati" (nested), cioè uno dentro l'altro. Dovete però stare attenti a rispettare le seguenti regole:

- Ogni istruzione FOR-TO-STEP deve essere accoppiata con una NEXT.
- Per diversi cicli annidati devono essere usate diverse *variabili di controllo*.
- Se un ciclo contiene una parte di un altro ciclo, deve contenerlo tutto.

In caso contrario, il calcolatore interromperà l'esecuzione del programma e stamperà il messaggio di errore "CAN'T DO THAT IN xx".

Esempi:

```
>NEW
>100 REM TROVARE IL PIU'
>101 REM PICCOLO NUMERO
>102 REM UGUALE ALLA SOMMA
>103 REM DEI CUBI DELLE
>104 REM SUE CIFRE
>110 FOR CENT=1 TO 9
>120 FOR DEC=0 TO 9
>130 FOR UNIT=0 TO 9
>140 S=100*CENT+10*DEC+UNI
 T
>150 IF S<>CENT^3+DEC^3+UN
 IT^3 THEN 180
 THEN 180
>160 PRINT S
>170 GOTO 210
>180 NEXT UNIT
>190 NEXT DEC
>200 NEXT CENT
>210 END
>RUN
  153

 ** DONE **
```

Potete uscire da un ciclo FOR-NEXT con la GOTO o la IF, ma non potete entrarci con queste istruzioni. Potete usare la GOSUB per uscire da un ciclo e ritornarvi. Assicuratevi di non usare la stessa *variabile di controllo* per un ciclo che si trovi nel sottoprogramma.

```
>NEW

>100 FOR I=1 TO 3
>110 PRINT I
>120 GOSUB 140
>130 NEXTI
>140 FOR I=1 TO 5
>150 PRINT I;
>160 NEXT I
>170 RETURN
>180 END
>RUN
  1
  1  2  3  4  5
 * CAN'T DO THAT IN 130
```

NEXT

NEXT *variabile di controllo*

L'istruzione NEXT è sempre accoppiata alla FOR-TO-STEP per la costruzione di un ciclo. La *variabile di controllo* è la stessa che compare nella corrispondente istruzione FOR-TO-STEP.

L'istruzione NEXT controlla se il calcolatore deve ripetere il ciclo o passare all'istruzione successiva.

Quando il calcolatore incontra l'istruzione NEXT, aggiunge l'*incremento* precedentemente calcolato alla *variabile di controllo*, e verifica se la *variabile di controllo* supera l'*estremo* specificato nella FOR-TO-STEP, se no, il ciclo viene ripetuto.

Esempi:

```
>NEW

>100 REM CONTO DA 1 A 10
>110 FOR X=1 TO 10
>120 PRINT X;
>130 NEXT X
>140 END
>RUN
 1  2  3  4  5  6  7  8  9
 10
  ** DONE **
```

```
>NEW

>100 REM CONTO ALLA ROVESCIA
>110 CALL CLEAR
>120 FOR I=10 TO 1 STEP -1
>130 PRINT I
>140 FOR T=1 TO 200
>150 NEXT T
>160 CALL CLEAR
>170 NEXT I
>180 PRINT "BASTA ORA"
>190 REM CAMBIO COLORE
>200 FOR C=2 TO 16 STEP 2
>210 CALL SCREEN(C)
>220 FOR T=1 TO 100
>230 NEXT T
>240 NEXT C
>250 END
>RUN

--APPARE IL CONTO ALLA
  ROVESCIA

 BASTA ORA

--LO SCHERMO CAMBIA COLORE

  8 VOLTE

 ** DONE **
```

Istruzioni di Ingresso-Uscita

Introduzione

Le istruzioni di INGRESSO-USCITA vi consentono di trasferire dati nel vostro programma e di ottenere i risultati. Questo paragrafo descrive queste istruzioni (PRINT, DISPLAY, INPUT, READ, DATA, RESTORE) e come usarle con la tastiera e il video del vostro calcolatore.

In un programma possono essere inseriti dati da tre tipi di sorgenti:

- dalla tastiera, con l'istruzione INPUT,
- dall'interno del programma, con le istruzioni READ, DATA e RESTORE
- da file registrati su dispositivi esterni, con l'istruzione INPUT

Possono essere emessi dati verso due tipi di dispositivi:

- il video, con le istruzioni PRINT o DISPLAY
- file registrati su dispositivi esterni, con l'istruzione PRINT (p.e. la stampante).

Ci sono altri due paragrafi, in questo libro, che descrivono le prestazioni di ingresso-uscita del vostro calcolatore. Il paragrafo "Gestione File" vi insegna a costruire le istruzioni per i dispositivi esterni. Vengono riportati diversi sottoprogrammi che eseguono operazioni di ingresso-uscita per la grafica, i colori e i suoni e il paragrafo "Grafica a Colori e Suoni" vi insegna a usarli.

INPUT

Risposta a un'istruzione di ingresso

Quando si esegue un'istruzione di INPUT da tastiera, si devono dare i valori corrispondenti alle variabili nello stesso ordine in cui queste compaiono nella INPUT. I valori devono essere dati tutti sulla stessa linea (fino a quattro linee del video), separati da virgole. I valori delle variabili di stringa possono essere dati tra virgolette; se però la stringa contiene una virgola, un apice, degli spazi prima o dopo, allora *deve* essere tra virgolette.

Esempi:

```
>NEW

>100 INPUT A$
>110 PRINT A$::
>120 INPUT B$
>130 PRINT B$::
>140 INPUT C$
>150 PRINT C$::
>160 INPUT D$
>170 X=500
>180 PRINT D$;X::
>190 INPUT E$
>200 PRINT E$
>210 END
>RUN
 ? "GIANNI, MARIA"
 GIANNI, MARIA

 ? """CIAO CARO"""
 "CIAO CARO"

 ? "GIANNI IL FIGLIO"
 GIANNI IL FIGLIO

 ? "PREZZO DI VENDITA: "

 PREZZO DI VENDITA: 500

 ? TEXAS
 TEXAS

 ** DONE **
```

I valori vengono assegnati alle variabili della *lista variabili* da sinistra verso destra. Quindi gli indici presenti nella *lista variabili* vengono calcolati solo dopo che è stato assegnato un valore alle variabili a sinistra.

In Extended BASIC le operazioni di ingresso si possono fare con l'istruzione LINPUT, mentre l'istruzione ACCEPT consente l'ingresso di dati da quasi tutte le posizioni del video.*

```
>NEW

>100 INPUT I,A(I)
>110 PRINT I:A(3)
>120 END
>RUN
 ? 3,7
  3
  7

 ** DONE **
```

INPUT

Ogni dato in ingresso viene controllato dal calcolatore se non è corretto, compare il messaggio "WARNING: INPUT ERROR, TRY AGAIN", e la linea va riscritta. Questo messaggio compare se:

- cercate di inserire dati in un numero minore o maggiore delle variabili richieste nell'istruzione INPUT;
- date un valore alfanumerico per una variabile numerica. (Ricordate, invece, che un numero può essere considerato variabile di stringa).

Esempi:

```
>NEW

>100 INPUT A,B$
>110 PRINT A;B$
>120 END
>RUN
 ? 12,HI,3

  * WARNING:
    INPUT ERROR IN 100

TRY AGAIN: HI,3

* WARNING:
  INPUT ERROR IN 100
TRY AGAIN: 23,HI
 23 HI

** DONE **
```

Se date in ingresso un numero che provoca un overflow, compare il messaggio "WARNING: NUMBER TOO BIG, TRY AGAIN", e dovete riscrivere la linea. Se invece date un numero che causa un underflow, il suo valore viene sostituito con zero, e non viene inviato alcun messaggio.

In Extended BASIC* la segnalazione degli errori è più precisa, e vi aiuta quindi a individuarli più rapidamente.

```
>NEW

>100 INPUT A
>110 PRINT A
>120 END
>RUN
 ? 23E139

  * WARNING:
    NUMBER TOO BIG IN 100

TRY AGAIN: 23E-139
0

** DONE **
```

* in vendita separatamente

READ

READ *lista-variabili*

L'istruzione READ serve per leggere dati registrati all'interno dello stesso programma in un'istruzione DATA. La *lista variabili* specifica le variabili a cui devono essere assegnati i valori. I nomi delle variabili nella *lista* sono separati da virgole, e possono essere variabili numeriche o di stringa.

Esempi:

```
>NEW

>100 FOR I=1 TO 3
>110 READ X,Y
>120 PRINT X;Y
>130 NEXT I
>140 DATA 2,5,8,36,2.3,32

>150 END
>RUN
  2  5
  8  36
  2.3  32
 ** DONE **
```

DATA

DATA *lista-dati*

L'istruzione DATA vi permette di registrare dei dati all'interno del vostro programma. Quando il programma va in esecuzione, i dati della *lista-dati* si leggono con un'istruzione READ, che li assegna alle variabili specificate nella sua *lista-variabili*. Gli elementi nella *lista-dati* sono separati da virgole. Quando un programma incontra un'istruzione DATA, procede alla successiva senza eseguire alcuna operazione.

Le istruzioni DATA possono essere in qualsiasi punto del programma, ma è importante l'ordine con cui si succedono. I dati della *lista-dati* vengono letti sequenzialmente, a partire dal primo elemento nella prima istruzione DATA. Se in un programma ci sono più istruzioni DATA, queste vengono lette in ordine di numero di linea crescente. Perciò l'ordine in cui compaiono i dati nelle *liste-dati* e l'ordine delle istruzioni DATA determina l'ordine in cui i dati vengono letti.

Esempi:

```
>NEW
>100 FOR I=1 TO 5
>110 READ A,B
>120 PRINT A;B
>130 NEXT I
>140 DATA 2,4,6,7,8
>150 DATA 1,2,3,4,5
>160 END
>RUN
   2  4
   6  7
   8  1
   2  3
   4  5

 ** DONE **
```

DATA

I dati nella *lista dati* devono corrispondere al tipo della variabile a cui vanno assegnati. Perciò se nella READ è specificata una variabile numerica, nella posizione corrispondente dell'istruzione DATA ci deve essere una costante numerica. Analogamente, se è specificata una variabile di stringa, nella posizione corrispondente della DATA ci deve essere una costante di stringa. Ricordate che un numero è una stringa valida.

Quando usate una costante di stringa in un'istruzione DATA, potete metterla tra virgolette; ma se contiene virgole, apici, spazi prima o dopo, *deve* essere tra virgolette.

Esempi:

```
>NEW

>100 READ A$,B$,C,D
>110 PRINT A$:B$:C:D
>120 DATA CIAO,"GIANNI, PIA"
>125 DATA 28,3.1416
>130 END
>RUN
 CIAO
 GIANNI, PIA
  28
  3.1416

  ** DONE **
```

Se nella *lista-dati* di un'istruzione DATA compaiono due virgole adiacenti, il calcolatore le interpreta come una variabile di stringa di lunghezza zero (una stringa senza caratteri). Nell'esempio a destra, alla linea 110 compaiono due virgole adiacenti, e la variabile corrispondente B$ viene considerata una stringa vuota.

```
>NEW

>100 READ A$,B$,C
>110 DATA HI,,2
>120 PRINT "A$: ";A$
>130 PRINT "B$: ";B$
>140 PRINT "C: ";C
>150 END
>RUN
 A$: HI

 B$:
 C:  2

 ** DONE **
```

Quando viene eseguita un'istruzione READ, se nella *lista variabili* ci sono più variabili di quante ne rimangano nella DATA, sul video compare il messaggio "DATA ERROR", e il programma si ferma. Se viene letta una costante numerica che causa un underflow, il suo valore viene sostituito con zero, non viene inviato alcun messaggio, e il programma continua normalmente la sua esecuzione. Se viene letta una costante che provoca un overflow, il suo valore viene sostituito con uno degli estremi previsti dal calcolatore, sul video compare il messaggio "WARNING:NUMBER TOO BIG", e il programma continua. Per informazioni su underflow, overflow, e limiti delle costanti numeriche, vedi il paragrafo "Costanti Numeriche".

```
>NEW

>100 READ A,B
>110 DATA 12E-135
>120 DATA 36E142
>130 PRINT :A:B
>140 READ C
>150 END
>RUN

 * WARNING:
   NUMBER TOO BIG IN 100


  0
 9.99999E+**

 * DATA ERROR IN 140

>▮
```

RESTORE

RESTORE [*numero-di-linea*]

(Vedere il paragrafo "Gestione File" per l'uso della RESTORE nella gestione file.)

L'istruzione RESTORE dice al programma quale istruzione DATA usare con la prossima istruzione READ.

RESTORE senza il *numero-di-linea* fa si che quando viene eseguita la READ successiva, i valori vengono assegnati partendo dalla prima istruzione DATA del programma.

Esempi:

```
>NEW

>100 FOR I=1 TO 5
>110 READ X
>120 RESTORE
>130 PRINT X;
>140 NEXT I
>150 DATA 10,20,30
>160 END
>RUN
  10 10 10 10 10

  ** DONE **
```

Quando invece è seguita dal *numero-di-linea* di un'istruzione DATA, all'esecuzione della READ successiva, i valori sono assegnati a partire dal primo elemento della DATA specificata dal *numero-di-linea.*

```
>NEW

>100 FOR I=1 TO 2
>110 FOR J=1 TO 4
>120 READ A
>130 PRINT A;
>140 NEXT J
>150 RESTORE 180
>160 NEXT I
>170 DATA 12,33,41,26,42
>175 DATA 50
>180 DATA 10,20,30,40,50
>190 END
>RUN
 12 33 41 26 10 20 30
 40

 ** DONE **
```

Nel programma sulla destra potete vedere come interagiscono le istruzioni READ, DATA e RESTORE. Nella linea 120 il calcolatore assegna ad A e B i valori presi dalla DATA con il numero di linea più basso, la linea 180. La prima READ, quindi, pone A=2 e B=4. La successiva esecuzione della READ prende i dati dalla linea 180 e pone A=6 e B=8. La terza esecuzione della READ assegna l'ultimo dato della linea 180 ad A, e il primo della 190 a B, e pone A=10 e B=12. Infine, la quarta esecuzione della READ, l'ultima del ciclo J, continua a prelevare dati dalla linea 190, e pone A=14 e B=16. Notate che prima di passare al successivo ciclo I, il calcolatore incontra un'istruzione RESTORE alla linea 160, che gli dice di prelevare i dati per la prossima READ dall'inizio della linea 190. Il programma continua quindi a leggere i dati dalle linee 190 e poi 200.

```
>NEW

>100 FOR I=1 TO 2
>110 FOR J=1 TO 4
>120 READ A,B
>130 PRINT A;B;
>140 NEXT J
>150 PRINT
>160 RESTORE 190
>170 NEXT I
>180 DATA 2,4,6,8,10
>190 DATA 12,14,16,18
>200 DATA 20,22,24,26
>210 END
>RUN

 2  4  6  8  10  12  14  16
 12  14  16  18  20  22  24

 26

 ** DONE **
```

Se il *numero-di-linea* specificato nell'istruzione RESTORE non corrisponde a un'istruzione DATA o non esiste nel programma, la successiva istruzione READ comincia a leggere dati dalla prima istruzione DATA con numero di linea maggiore o uguale a quello specificato. Se non ce n'è, compare il messaggio "DATA ERROR" Se il *numero-di-linea* specificato è maggiore del più alto numero di linea del programma, il programma si ferma e compare il messaggio "DATA ERROR IN xx".

PRINT

PRINT [*lista di stampa*]

(Le istruzioni per l'uso della PRINT con i file sono nel paragrafo "Gestione File".)

L'istruzione PRINT vi consente di stampare numeri e stringhe sul video. La *lista di stampa* è costituita da

- *elementi di stampa* - espressioni numeriche e di stringa, i cui risultati compaiono sul video, e *funzioni-tab* che controllano le posizioni di stampa (come il tasto tabulatore di una macchina da scrivere).
- *separatori* - segni di punteggiatura tra i diversi *elementi di stampa*, che servono da indicatori della posizione di stampa.

Quando il calcolatore esegue un'istruzione di PRINT, i valori delle espressioni della *lista di stampa* compaiono sul video nell'ordine da sinistra a destra, nelle posizioni definite dai *separatori* e dalle *funzioni-tab*.

Esempi:

```
>NEW
>100 A=10
>110 B=20
>120 S$="TI COMPUTER"
>130 PRINT A;B:S$
>140 PRINT "CIAO AMICI"

>150 END
>RUN
  10  20

 TI COMPUTER
 CIAO AMICI

 ** DONE **
```

Stampa di Stringhe

Le espressioni di stringa vengono calcolate e producono un risultato alfanumerico che viene stampato senza spazi prima e dopo. Se volete stampare degli spazi, potete inserirli nella stringa stessa, o metterli separatamente tra virgolette.

```
>NEW

>100 N$="GIANNI"
>110 M$="HI"
>120 PRINT M$;N$
>130 PRINT M$&" "&N$
>140 PRINT "CIAO ";N$
>150 END
>RUN
 HIGIANNI
 HI GIANNI
 CIAO GIANNI

 ** DONE **
```

Stampa di Numeri

Le espressioni numeriche vengono calcolate e il loro risultato è stampato. I numeri positivi vengono stampati con uno spazio davanti (al posto del segno più), e i numeri negativi con un segno meno. Tutti i numeri vengono separati da uno spazio.

In Extended BASIC esiste l'istruzione PRINT...USING con cui potete controllare il formato della stampa su video.

```
>NEW

>100 LET A=10.2
>110 B=-30.5
>120 C=16.7
>130 PRINT A;B;C
>140 PRINT A+B
>150 END
>RUN
  10.2 -30.5  16.7

 -20.3

 ** DONE **
```

* in vendita separatamente

PRINT

La Funzione Tab

La *funzione tab* ha lo stesso effetto del tasto tabulatore di una macchina da scrivere, e serve per posizionare le stampe.

TAB (*espressione numerica*)

Dando il numero di colonna *n* si definisce la posizione sulla linea del video in cui si vuole stampare il carattere successivo. L'*espressione numerica* viene calcolata e il risultato è arrotondato all'intero più vicino *n*.

Se *n* è minore di uno, il suo valore viene sostituito con uno. Se è maggiore di 28, gli viene sottratto 28 tante volte quante sono necessarie per ottenere $1 \leqslant n \leq 28$. Se il numero di caratteri già stampati sulla linea è minore o uguale a *n*, il successivo *elemento di stampa* è stampato a partire dalla posizione *n*. Se invece il numero di caratteri già stampati sulla linea è maggiore di *n*, l'elemento successivo viene stampato a partire dalla posizione *n*, ma nella linea seguente. Notate che la *funzione tab* è un *elemento di stampa*. Deve quindi essere preceduta da un *separatore*, se non è il primo elemento della *lista di stampa*, mentre deve essere seguita da un *separatore* se non è l'ultimo elemento. Il *separatore* che si trova prima di un *funzione tab* viene calcolato prima della funzione stessa. Quello che si trova dopo, dopo la funzione. Perciò vi conviene usare il *punto e virgola* come *separatore* prima e dopo la *funzione tab* per avere i migliori risultati.

Il programma sulla destra esegue le operazioni che seguono.

- linea 120 - stampa A, si sposta alla posizione 15 e stampa B
- linea 130 - stampa A, si sposta alla successiva zona di stampa (in questo caso la quindicesima della stessa linea), e stampa B.
- linea 140 - stampa A, si sposta alla posizione 15 come specificato dalla *funzione tab*, poi si sposta alla successiva posizione di stampa come indicato dalla virgola (in questo caso la posizione 1 della linea successiva), e stampa B
- linea 150 - si sposta alla posizione 5, stampa A, si sposta alla posizione 6 della linea successiva (in quanto la posizione 6 della linea in corso era già stata superata con la stampa di A), e stampa B
- linea 160 - stampa A, sottrae 28 da 43 per dare alla *funzione tab* un valore valido, si sposta alla posizione 15 (43-28=15), e stampa B

Esempi:

```
>NEW

>100 A=326
>110 B=79
>120 PRINT A;TAB(15);B
>130 PRINT A,B
>140 PRINT A;TAB(15),B
>150 PRINT TAB(5);A;TAB(6);B
>160 PRINT A;TAB(43);B
>170 END

>RUN
  326            79
  326            79
  326
  79
      326
       79
  326            79

  ** DONE **
```

Separatori

Per l'istruzione PRINT ogni linea del video ha 28 posizioni numerate da sinistra verso destra (1-28). Ogni linea è divisa in due zone di stampa di 14 caratteri. Con i *separatori* e la *funzione tab* si può controllare la posizione dei caratteri sul video.

Ci sono tre tipi di *separatori*, punto e virgola, due punti e virgola. Ci deve essere almeno uno di questi tra due *elementi di stampa* nella *lista di stampa*. Mettendo più *separatori* di seguito, essi vengono calcolati da sinistra a destra.

Il *punto e virgola* fra due *elementi* fa sì che siano stampati uno di fianco all'altro senza spazi intermedi. I numeri vengono stampati con uno spazio davanti, indipendentemente dal *separatore* usato.

Esempi:

```
>NEW

>100 A=-26
>110 B=-33
>120 C$="CIAO"
>130 D$="COME STAI?"
>140 PRINT A;B;C$;D$
>150 END
>RUN
 -26 -33 CIAOCOME STAI?

  ** DONE **
```

I *due punti* servono per stampare l'*elemento* successivo all'inizio della linea successiva.

```
>NEW

>100 A=-26
>110 B$="CIAO"
>120 C$="COME STAI?"
>130 PRINT A$:B$:C$
>140 END
>RUN
 -26
 CIAO

 COME STAI?

 ** DONE **
```

Le linee di stampa sono divise in due zone, la prima dalla colonna 1, e la seconda dalla 15. Un *elemento* preceduto dalla *virgola* viene stampato all'inizio della zona successiva. Se la stampa si trova già nella seconda zona di una linea, la *virgola* la fa passare alla linea successiva.

```
>NEW

>100 A$="ZONA 1"
>110 B$="ZONA 2"
>120 PRINT A$,B$
>130 PRINT A$:,B$,A$
>140 END
>RUN
 ZONA 1        ZONA 2
 ZONA 1
               ZONA 2
 ZONA 1

 ** DONE **
```

DISPLAY

DISPLAY [*lista di stampa*]

L'istruzione DISPLAY può essere usata solo con il video, ed è identica alla PRINT. Le regole che segue sono le stesse della PRINT.

In Extended BASIC* esiste anche la DISPLAY ... USING.

Esempi:

```
>NEW

>100 B$="HIHI"
>110 A=35.6
>120 C=49.7
>130 PRINT B$:A;C
>140 DISPLAY B$:A;C
>150 END
>RUN
 HIHI

  35.6  49.7
 HIHI
  35.6  49.7

 ** DONE **
```

Grafica a Colori e Suoni

Introduzione

Il vostro calcolatore ha una serie di sottoprogrammi per la grafica a colori, i suoni e altre prestazioni, in genere non disponibili in BASIC.

Quando ne volete usare uno, lo *chiamate* col suo nome e fornite alcune specifiche. Il programma va in esecuzione dandovi la possibilità di produrre musiche, colori e dei particolari caratteri grafici. Tutti questi sottoprogrammi possono essere usati in Modo Comandi o richiamati da programma.

A seconda delle operazioni che eseguono, si suddividono in:

- sottoprogrammi di INGRESSO - GCHAR, JOYST, KEY
- sottoprogrammi di USCITA - CLEAR, HCHAR, VCHAR, SOUND, SCREEN
- sottoprogrammi INTERNI - CHAR, COLOR (i risultati di questi ultimi non si vedono finchè non usate un'istruzione di USCITA per visualizzarli sul video).

In Extended BASIC è una scelta ampia di sottoprogrammi di questo tipo, che rendono più facile la programmazione e aumentano le prestazioni del calcolatore (CHAR PAT, CHAR SET, COINC, LOCATE, MAGNIFY, MOTION, PATTERN, SAY**, SPRITE, ecc.)*

* in vendita separatamente
** in conbinazione con il sintetizzatore vocale Texas Instruments

Sottoprogramma CLEAR

CALL CLEAR

Serve per cancellare il video; quando viene chiamato, riempie tutte le posizioni di spazi (codice ASCII 32).

Esempi:

```
>PRINT "CIAO CIAO"
 CIAO CIAO
>CALL CLEAR

--LO SCHERMO SI PULISCE
```

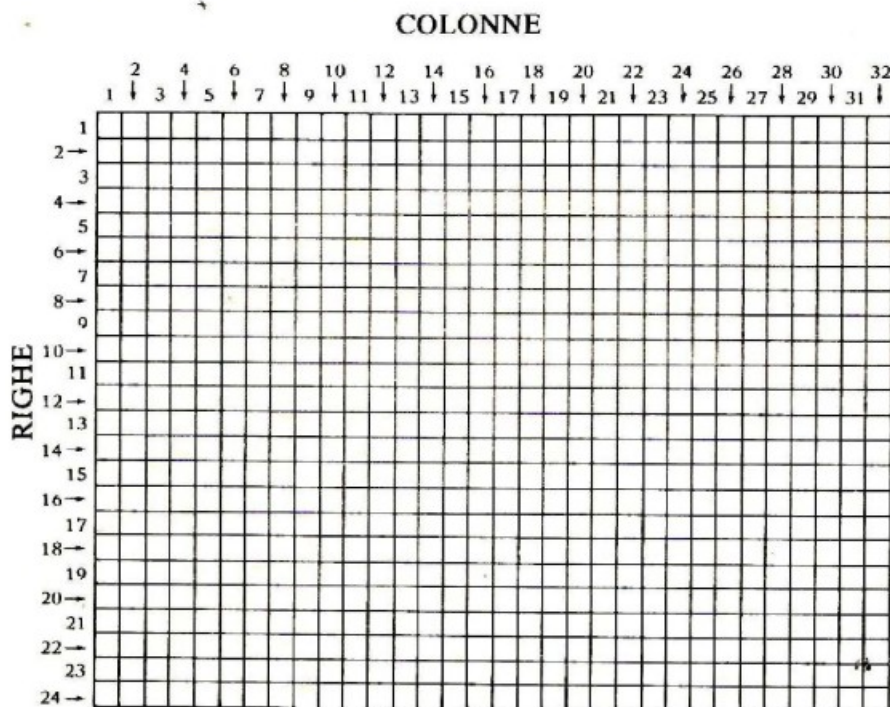
Quando il programma sulla destra viene eseguito, il video viene cancellato prima dell'istruzione PRINT.

```
>NEW
>100 CALL CLEAR
>110 PRINT "CIAO CIAO"
>120 PRINT "COME STAI?"

>130 END
>RUN

--LO SCHERMO SI PULISCE

 CIAO CIAO
 COME STAI?

  ** DONE **
```

Sottoprogramma HCHAR

(Ripetizione Orizzontale di Caratteri - Horizontal CHAracter Repetition)

CALL HCHAR (*numero di riga, numero di colonna, codice carattere* [, *numero di ripetizioni*])

Il sottoprogramma HCHAR stampa un carattere in qualsiasi posizione del video, ed eventualmente lo ripete sulla stessa linea. Il *numero di riga* e il *numero di colonna* individua la posizione di partenza sul video. Il *numero di riga*, il *numero di colonna*, il *codice carattere* e il *numero di ripetizioni* sono espressioni numeriche.

Se il risultato di una di queste espressioni numeriche non è intero, viene arrotondato. I limiti per ognuno dei parametri sono:

Valore	*limiti*
Numero di riga	*1-24 compresi*
Numero di colonna	*1-32 compresi*
Codice carattere	*0-32767 compresi*
Numero ripetizioni	*0-32767 compresi*

Poichè le colonne 1, 2, 31 e 32 non si vedono sul video, potete usare solo *numeri di colonna* da 3 a 30.

Come *codice carattere* potete dare qualsiasi numero fino a 32767, ma il calcolatore lo convertirà in un valore compreso tra 0 e 255. I codici che vanno da 32 a 127 sono quelli standard ASCII, quelli da 128 a 159 possono essere definiti con il sottoprogramma CHAR. Se date un *codice carattere* che non corrisponde ad alcun carattere , otterrete quello che si trova in memoria al momento in cui chiamate il sottoprogramma HCHAR.

Esempi:

```
>CALL CLEAR

--LO SCHERMO SI PULISCE

>CALL HCHAR(10,1,72,50)
```

```
HHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH
HHHHHHHHHHHHHHHHHH



>CALL HCHAR(10,1,72,50)
```

```
>NEW

>100 CALL CLEAR
>110 FOR I=9 TO 15
>120 CALL HCHAR(I,13,36,6)
>130 NEXT I
>140 GOTO 140
>RUN

--LO SCHERMO SI PULISCE
```

```
$$$$$$
$$$$$$
$$$$$$
$$$$$$
$$$$$$
$$$$$$
$$$$$$
```

```
>CALL HCHAR(24,14,29752)

              8
>CALL HCHAR(24,14,35)
              #
>CALL HCHAR(24,14,132)

--IL CARATTERE MOSTRATO

  DIPENDE DAL CONTENUTO

  ATTUALE DELLA MEMORIA
```

Sottoprogramma VCHAR

(Ripetizione Verticale di Caratteri - Vertical CHAracter Repetition)

CALL VCHAR (*numero di riga, numero di colonna, codice carattere.* [, *numero di ripetizioni*])

Il sottoprogramma VCHAR funziona come HCHAR, salvo che i caratteri vengono ripetuti verticalmente invece che orizzontalmente. Il calcolatore stampa sul video il carattere a partire dalla posizione specificata verso il basso del video. Se raggiunge il fondo continua a partire dalla cima della colonna successiva. Se raggiunge il bordo destro del video, continua su quello sinistro. Vi rimandiamo al sottoprogramma HCHAR per ulteriori dettagli.

Esempi:

```
>NEW

>100 CALL CLEAR
>110 FOR I=13 TO 18
>120 CALL VCHAR(9,I,36,6)

>130 NEXT I
>140 GOTO 140
>RUN

--LO SCHERMO SI PULISCE
```

Sottoprogramma GCHAR

(Acquisizione Caratteri - Get CHARacter)

CALL GCHAR (*numero di riga, numero di colonna, variabile numerica*)

Il sottoprogramma GCHAR serve per leggere un carattere da qualsiasi posizione del video, individuata dal *numero di riga* e dal *numero di colonna*. Il calcolatore copia il codice ASCII del carattere letto nella *variabile numerica*.

Il *numero di riga* e il *numero di colonna* sono espressioni numeriche. Se il valore risultante non è intero, viene arrotondato in modo da diventarlo. Un *numero di riga* 1 indica la linea più alta del video, un *numero di colonna* 1 indica la prima colonna sulla sinistra.

Esempi:

```
>NEW

>100 CALL CLEAR
>110 CALL HCHAR(1,1,36,768)

>120 CALL GCHAR(5,10,X)

>130 CALL CLEAR
>140 PRINT X
>150 END
>RUN

--LO SCHERMO SI PULISCE

--LO SCHERMO SI RIEMPIE

  CON IL CAR. $ (COD.36)

--LO SCHERMO SI PULISCE

 36

 ** DONE **
```

Sottoprogramma COLOR

CALL COLOR (*numero dell'insieme di caratteri, codice colore del carattere, codice del colore di sfondo*)

Il sottoprogramma COLOR vi dà la possibilità di eseguire disegni sul video, specificando i colori per i diversi caratteri. (Per cambiare il colore del video stesso, vedi il sottoprogramma SCREEN.) I parametri tra parentesi sono tutti espressioni numeriche.

Ogni carattere visualizzato usa due colori: il colore dei punti che formano il carattere stesso si chiama *foreground color* (codice colore del carattere), quello che occupa la parte restante *background color* (colore dello sfondo). Sono disponibili 16 colori, numerati da 1 a 16 come si vede dalla tabella seguente:

Codice-colore	*Colore*
1	Trasparente
2	Nero
3	Verde
4	Verde chiaro
5	Blu scuro
6	Blu chiaro
7	Rosso scuro
8	Viola
9	Rosso
10	Rosso chiaro
11	Giallo scuro
12	Giallo chiaro
13	Verde scuro
14	Magenta
15	Grigio
16	Bianco

Se è specificato il colore trasparente (codice 1), attraverso il carattere visualizzato si vede il colore del video. Il *foreground color* standard è nero (codice 2), e il *background color* standard è trasparente (codice 1) per tutti i caratteri. Tutte le volte che si incontra un punto di arresto i caratteri tornano ai colori standard.

Esempi:

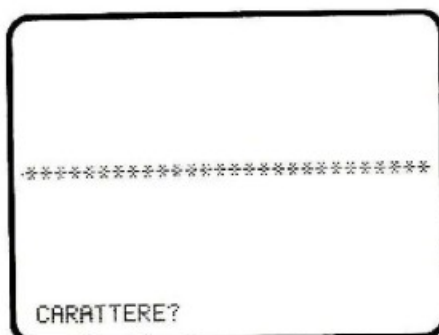
```
>NEW
>100 CALL CLEAR
>105 PRINT "SCELTA COLORI"
>110 INPUT "CARATTERE?":C
>120 INPUT "SFONDO?":S
>130 CALL CLEAR
>140 CALL COLOR(E,C,S)
>150 CALL HCHAR(12,3,42,28)
>160 GOTO 110
>RUN
```

--LO SCHERMO SI PULISCE

```
CARATTERE?2
SFONDO?14
```

--LO SCHERMO SI PULISCE

--ESCONO 28 ASTERISCHI

NERI SU SFONDO MAGENTA

PREMERE CLEAR PER FERMARE IL PROGRAMMA

```
>NEW
>100 CALL CLEAR
>110 CALL COLOR(2,1,7)
>120 CALL HCHAR(12,3,42,28)
>130 GOTO 140
>RUN
```

--LO SCHERMO SI PULISCE
--ESCONO 28 ASTERISCHI

NERO TRASPARENTI SU SFONDO ROSSO SCURO

SUL VIDEO VERDE

--PREMETE CLEAR PER FERMARE

Sottoprogramma COLOR

Nell'usare la CALL COLOR dovete specificare a quale dei sedici insiemi di caratteri appartiene quello che state stampando. L'elenco dei codici ASCII dei caratteri standard è più avanti, sotto la funzione ASC. Il carattere viene visualizzato nel colore desiderato quando si usano le routine HCHAR e VCHAR. Diamo di seguito la tabella degli *insiemi dei caratteri.*

Numero Insieme	*Codici Carattere*
1	32-39
2	40-47
3	48-55
4	56-63
5	64-71
6	72-79
7	80-87
8	88-95
9	96-103
10	104-111
11	112-119
12	120-127
13	128-135
14	136-143
15	144-151
16	152-159

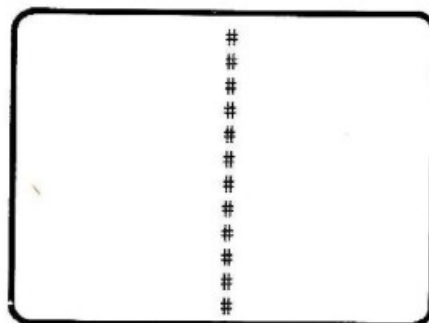
Notate che le 24 righe e le 32 colonne del video sono tutte riempite con degli spazi, finchè non mettete altri caratteri in qualche posizione. Se usate l'insieme dei caratteri 1 nell'istruzione CALL COLOR tutti gli spazi del video assumono il *background color* specificato, poichè il codice dello spazio appartiene a questo insieme. Potete verificare ciò con il programma sulla destra.

Esempi:

```
>NEW
>100 CALL CLEAR
>110 CALL COLOR(1,16,14)
>120 CALL VCHAR(1,15,35,24)
>130 GOTO 130
>RUN
```

--LO SCHERMO SI PULISCE

--ESCONO 24 # BIANCHI TRASPARENTI SU SFONDO

MAGENTA SUL VIDEO VERDE

--IL COLORE DELLO SCHERMO APPARE SOLO NELLA PARTE SUPERIORE ED INFERIORE

PREMETE CLEAR PER FERMARE IL PROGRAMMA

Sottoprogramma SCREEN

CALL SCREEN (*codice colore*)

Il sottoprogramma SCREEN aumenta le capacità grafiche del vostro calcolatore, permettendovi di cambiare il colore del video. Il colore standard del video durante l'esecuzione di un programma è verde chiaro (*codice colore* = 4).

Il *codice colore* è un'espressione numerica, il cui valore deve essere compreso tra 1 e 16, come si vede nella tabella del paragrafo CALL COLOR

Quando viene eseguita la CALL SCREEN, l'intero video assume il colore specificato dal *codice colore*. I caratteri sul video restano gli stessi, a meno che siano in trasparente foreground o background. In questo caso lasciano intravedere il nuovo colore del video.

Quando un programma si ferma perchè incontra un breakpoint (punto di arresto), o perchè è finito, il video assume il colore viola (codice 8). Se viene fatto ripartire, il video ritorna al colore standard (verde chiaro).

Esempi:

```
>NEW

>100 CALL CLEAR
>110 PRINT "SCELTA COLORI"
>115 INPUT "SCHERMO?":V
>120 INPUT "CARATTERE?":C
>130 INPUT "SFONDO?":S
>140 CALL CLEAR
>150 CALL SCREEN(V)
>160 CALL COLOR(2,C,S)
>170 CALL HCHAR(12,3,42,28)

>180 GOTO 110
>RUN

--LO SCHERMO SI PULISCE

 SCELTA COLORI
 SCHERMO?7
 CARATTERE?13
 SFONDO?16

--LO SCHERMO SI PULISCE

--ESCONO 28 ASTERISCHI
  VERDE SCURO CON SFONDO
  BIANCO SU SCHERMO ROSSO
  SCURO
```

```
****************************

SCELTA COLORI
SCHERMO?
```

Sottoprogramma CHAR

(Definizione carattere - CHARacter definition)

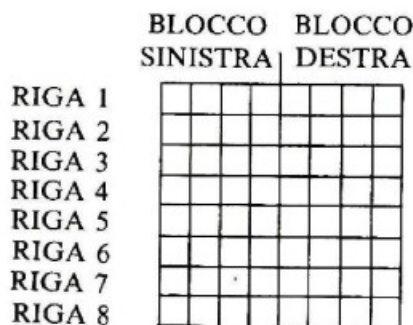
CALL CHAR *(codice carattere, "identificatore di sagoma")*

Il sottoprogramma CHAR vi consente di definire dei vostri speciali caratteri grafici. Potete ridefinire l'insieme dei caratteri standard (codici ASCII 32-127) e definirne degli altri con codici 128-159.

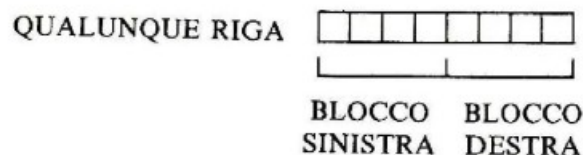
Il *codice carattere* specifica il codice del carattere che volete definire, e deve essere un'espressione numerica con valore da 32 a 159 compresi. Se è tra 128 e 159 e non c'è abbastanza memoria libera per definire il carattere, il programma si ferma inviando il messaggio di errore "MEMORY FULL".

L'*identificatore di sagoma* è un'espressione di stringa di 16 caratteri che specifica la forma del vostro carattere, con una rappresentazione dei 64 punti che costituiscono una posizione del video. Questi 64 punti formano una matrice come quella della figura seguente.

BLOCCO SINISTRA | BLOCCO DESTRA

RIGA 1								
RIGA 2								
RIGA 3								
RIGA 4								
RIGA 5								
RIGA 6								
RIGA 7								
RIGA 8								

Ogni riga è suddivisa in due blocchi di quattro punti ognuno:

QUALUNQUE RIGA								

BLOCCO SINISTRA — BLOCCO DESTRA

Esempi:

```
>NEW

>100 CALL CLEAR
>110 CALL CHAR(33,"FFFFFFFFFF
 FFFFFF")
>120 CALL COLOR(1,9,6)
>130 CALL VCHAR(12,16,33)
>140 GOTO 140
>RUN

--LO SCHERMO SI PULISCE ■
```

Sottoprogramma CHAR

Ogni carattere dell'espressione di stringa definisce la distribuzione dei punti in un blocco di una riga. Le righe sono descritte da sinistra verso destra e dall'alto al basso. i primi due caratteri della stringa descrivono la distribuzione dei punti nella prima riga della matrice, gli altri due la seconda riga, e così via.

I caratteri vengono creati "accendendo"alcuni punti e "spegnendone" altri. Lo spazio, ad esempio, (codice 32) ha tutti i punti "spenti"; il carattere (■) ha tutti i punti "accesi".

I caratteri standard sono definiti automaticamente in modo da "accendere" i punti opportuni. Per creare un nuovo carattere, dovete dire al calcolatore quali punti "accendere" e quali "spegnere" in ciascuno dei sedici blocchi che contengono il carattere; per questo si usa un codice binario, tradotto in un codice (esadecimale) più facile da manipolare, costituito da numeri e da lettere. La tabella seguente riporta i codici per ogni possibile distribuzione di punti in un blocco di una riga.

Blocchi	*Codice Binario (0 = Off; 1 = On)*	*Codice Esadecimale*
□□□□	0000	0
□□□■	0001	1
□□■□	0010	2
□□■■	0011	3
□■□□	0100	4
□■□■	0101	5
□■■□	0110	6
□■■■	0111	7
■□□□	1000	8
■□□■	1001	9
■□■□	1010	A
■□■■	1011	B
■■□□	1100	C
■■□■	1101	D
■■■□	1110	E
■■■■	1111	F

Nota: i codici esadecimali A, B, C, D, E e F devono essere dati come caratteri maiuscoli.

Sottoprogramma CHAR

Per descrivere la sagoma della figura seguente dovreste usare la stringa "1898FF3D3C3CE404"

BLOCCHI DI SINISTRA BLOCCHI DI DESTRA CODICI BLOCCO

Se l'espressione di stringa ha meno di 16 caratteri, il calcolatore considera 0 quelli che mancano; se ha più di 16 caratteri vengono ignorati.

Ricordate che la CALL CHAR serve solo per definire un carattere; per visualizzarlo dovete usare CALL HCHAR, CALL VCHAR, PRINT, o DISPLAY. Quando viene eseguita la CALL CHAR, ogni carattere già presente sul video con quel *codice* viene cambiato nel nuovo carattere.

Esempi:

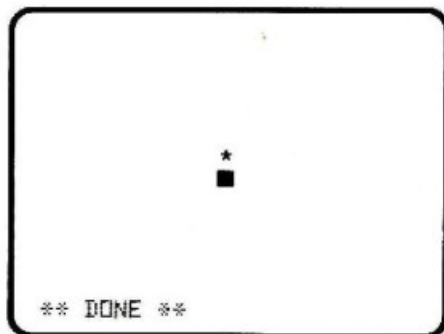
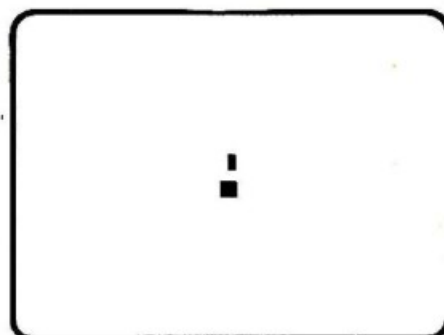
```
>NEW

>100 CALL CLEAR
>110 A$="1898FF3D3C3CE404"
>120 B$="1819FFBC3C3C2720"
>130 CALL CHAR(128,A$)
>140 CALL CHAR(129,B$)
>150 CALL COLOR(9,7,12)
>160 CALL VCHAR(12,16,128)
>170 FOR T=1 TO 500
>180 NEXT T
>190 CALL VCHAR(12,16,129)
>200 FOR T=1 TO 500
>210 NEXT T
>220 GOTO 140
>230 END
>RUN

--LO SCHERMO SI PULISCE

--IL CARATTERE SI MUOVE

--PREMETE CLEAR PER FERMARE
  IL PROGRAMMA
```

```
>NEW
>100 CALL CLEAR
>110 CALL CHAR(128,"0103070F1
 F3F7FFF")
>120 PRINT CHR$(128)
>130 END
>RUN

--LO SCHERMO SI PULISCE

** DONE **
```

Sottoprogramma CHAR

Se un programma si ferma per un breakpoint, i caratteri con codici 32-127 ritornano alla loro normale rappresentazione. Quelli con codici 128-159 restano invariati. Se invece si ferma perchè è finito o perchè incontra un errore tutti i caratterei da 32 a 127 tornano alla normale rappresentazione, e gli altri (128-159) restano non definiti.

Esempi:

```
>NEW

>100 CALL CLEAR
>110 CALL CHAR(128,"FFFFFFFFF
 FFFFFFF")
>120 CALL CHAR(42,"0F0F0F0F0F
 0F0F0F")
>130 CALL HCHAR(12,17,42)
>140 CALL HCHAR(14,17,128)
>150 FOR T=1 TO 350
>160 NEXT T
>170 END

>RUN

--LO SCHERMO SI PULISCE
```

```
>CALL HCHAR(24,5,42)
      *
```

Sottoprogramma SOUND

CALL SOUND (*durata, frequenza 1, volume 1* [, *frequenza 2, volume 2*] [, *frequenza 3, volume 3*] [, *frequenza 4, volume 4*])

Il sottoprogramma SOUND dice al calcolatore di produrre suoni a diversa frequenza, specificando tre caratteristiche di ogni suono

- *durata* - quanto è lungo
- *frequenza* - la nota
- *volume* - quanto è forte

La *durata*, la *frequenza* e il *volume* sono espressioni numeriche, il cui risultato è intero, o viene arrotondato a un intero. Gli estremi validi per ognuno di questi valori sono dati nella tabella seguente, e spiegati successivamente.

Valore	*Estremi*
durata	da 1 a 4250 compresi
	da -1 a -4250 compresi
frequenza	(Tono) da 110 a 44733 compresi
	(Rumore) da -1 a -8 compresi
volume	da 0 (il più basso) a 30 (il più alto) compresi

Durata

La *durata* viene data in millisecondi. Un secondo è uguale a 1000 millisecondi, quindi la durata varia da 0,001 a 4,25 secondi. La durata reale può differire al massimo di 1/60 di secondo, e viene assegnata ad ogni suono generato da una particolare CALL SOUND

Il calcolatore continua a eseguire le istruzioni di un programma mentre viene prodotto un suono. Quando chiamate un sottoprogramma SOUND, aspetta che il suono precedente sia terminato, prima di eseguire una nuova CALL SOUND, a meno che sia stata data una *durata* negativa. In questo caso il suono viene interrotto , e comincia immediatamente il successivo.

Esempi:

```
>CALL SOUND(100,294,2)

--EMETTE UNA NOTA
```

```
>NEW

>100 TONO=110
>110 FOR C=1 TO 10
>120 CALL SOUND(-500,TONO,1)
>130 TONO=TONO+110
>140 NEXT C
>150 END
>RUN

--EMETTE 10 NOTE VELOCEMENTE

 ** DONE **

>120 CALL SOUND(+500,TONO,1)
>RUN
--EMETTE 10 NOTE LENTAMENTE

 ** DONE **
```

Sottoprogramma SOUND

Frequenza

Potete dare la frequenza di un suono o di un rumore. La frequenza di un suono va da 110 Hertz (un ciclo al secondo, Hz) a 44733Hz, molto al di sopra della soglia di udibilità umana (La frequenza prodotta realmente può differire dallo 0 al massimo del 10%, a seconda del suo valore). Le frequenze per le note musicali più comuni sono date nella Tabella "Frequenze delle note musicali".

Esempi:

```
>CALL SOUND(1000,440,2)

--EMETTE UNA NOTA

>CALL SOUND(500,-1,2)

--EMETTE UN RUMORE

>NEW
```

FREQUENZE DELLE NOTE MUSICALI

La seguente tabella da le frequenze, intere o arrotondate, di quattro ottave della scala temperata (con le note intervallate da un semitono). Anche se questo elenco non rappresenta tutta la gamma dei suoni, e neanche delle note, può essere utile programmare qualche motivo musicale.

Frequenza	*Nota*		*Frequenza*	*Nota*	
110	LA		440	LA	(un'ottava sopra DO centrale)
117	LA #, SI♭		466	LA #, SI♭	
123	SI		494	SI	
131	DO	(un'ottava sotto DO centrale)	523	DO	(un'ottava sopra DO centrale)
139	DO #, RE♭		534	DO #, RE♭	
147	RE		587	RE	
156	RE #, MI♭		622	RE #, MI♭	
165	MI		659	MI	
175	FA		698	FA	
185	FA #, SOL♭		740	FA #, SOL♭	
196	SOL		784	SOL	
208	SOL #, LA♭		831	SOL #, LA♭	
220	LA	(un'ottava sotto DO centrale)	880	LA	(un'ottava sopra LA corista)
220	LA	(un'ottava sotto DO centrale)	880	LA	(un'ottava sopra LA corista)
233	LA #, SI♭		932	LA #, SI♭	
247	SI		988	SI	
262	DO	(DO centrale)	1047	DO	
277	DO #, RE♭		1109	DO #, RE♭	
294	RE		1175	RE	
311	RE #, MI♭		1245	RE #, MI♭	
330	MI		1319	MI	
349	FA		1397	FA	
370	FA #, SOL♭		1480	FA #, SOL♭	
392	SOL		1568	SOL	
415	SOL #, LA♭		1661	SOL #, LA♭	
440	LA	(un ottava sopra DO centrale)	1760	LA	

Sottoprogramma SOUND

Se viene dato un valore negativo per la *frequenza*, viene prodotto un rumore invece che un suono. Il rumore può essere "bianco" o "periodico". La tabella seguente dà i rumori associati ai diversi valori della frequenza. Provate a produrli per cominciare a conoscerli.

Caratteristiche dei Rumori

Valore della Frequenza	Caratteristiche
-1	"Rumore Periodico" di Tipo 1
-2	"Rumore Periodico" di Tipo 2
-3	"Rumore Periodico" di Tipo 3
-4	"Rumore Periodico" che varia con la frequenza del terzo suono specificato
-5	"Rumore Bianco" di Tipo 1
-6	"Rumore Bianco" di Tipo 2
-7	"Rumore Bianco" di Tipo 3
-8	"Rumore Bianco" che varia con la frequenza del terzo suono specificato

Possono essere attivati al massimo tre suoni e un disturbo simultaneamente, e per ognuno di essi deve essere dato subito dopo il volume.

Esempi:

```
>NEW

>100 FOR N=-1 TO -8 STEP -1
>110 CALL SOUND(1000,N,2)
>120 NEXT N
>130 END
>RUN

 --GENERA 8 RUMORI DIVERSI

 ** DONE **
```

```
>CALL SOUND(2000,-3,5)

--GENERA UN RUMORE
```

```
>CALL SOUND(2500,440,2,659,5,
 880,10,-6,15)

--EMETTE 3 NOTE E 1 RUMORE
```

```
>DUR=2500
>VOL=2
>C=262
>E=330
>G=392
>CALL SOUND(DUR,C,VOL,E,VOL,G
 ,VOL)

--PRODUCE UN ACCORDO
```

Sottoprogramma KEY

CALL KEY (*unità di tastiera, variabile di ritorno, variabile di stato*)

Il sottoprogramma KEY vi consente di trasferire un carattere dalla tastiera direttamente al vostro programma, senza bisogno dell'istruzione INPUT e con un sensibile risparmio di tempo.
Le informazioni presenti sul video non vengono disturbate dall'esecuzione di una CALL KEY, in quanto il carattere corrispondente al tasto premuto non viene visualizzato sullo schermo. La *unità di tastiera*, che indica la tastiera usata come dispositivo di ingresso, è un'espressione numerica che puo avere un valore da 0 a 5, come segue:

- 0 = tastiera della console, nel modo specificato dalla CALL KEY
- 1 = parte sinistra della tastiera della console o controllo a distanza 1
- 2 = parte destra della tastiera della console o controllo a distanza 2
- 3,4,5 = modi particolari per la tastiera della console

L'*unità di tastiera* 0 ridefinisce la tastiera nel modo specificato nella precedente CALL KEY.

Le unità 1 e 2 sono usate quando si vuole dividere la tastiera in due parti uguali ma piu ' piccole, o quando si usano i bottoni di fuoco dei controlli a distanza.

Le *unità* 3, 4 e 5 ridefiniscono la tastiera per operazioni particolari. Il modo di funzionamento specificato determina i codici carattere prodotti da alcuni tasti.

L'*unità* 3 definisce il modo di funzionamento con la tastiera standard del TI-99/4A. (La maggior parte dei Moduli di Comando software usa questo modo.) In questo modo i caratteri alfabetici sia minuscoli che maiuscoli vengono trasformati in maiuscoli, e i tasti di funzione (**BACK,BEGIN-,CLEAR**,etc.) producono codici tra 1 e 15. Non sono attivati i caratteri di controllo.

L'*unità* 4 ridefinisce la tastiera in Modo Pascal. In questo modo i caratteri alfabetici sono sia maiuscoli che minuscoli, e i tasti di funzione producono codici da 129 a 143. I codici dei caratteri di controllo vanno da 1 a 31.

L'*unità* 5 ridefinisce la tastiera in Modo BASIC. I caratteri alfabetici sono sia maiuscoli che minuscoli, i codici di funzione sono da 1 a 15, e quelli di controllo da 128 a 159 (e 187).

La *variabile di ritorno* deve essere una variabile numerica, e contiene il codice numerico del carattere corrispondente al tasto premuto. Se l'unità usata è la tastiera (unità 0), i codici dei caratteri sono quelli ASCII, e vanno da 0 a 127. Con le unità 1 e 2, i codici dei caratteri vanno da 0 a 19.

La *variabile di stato* è una variabile numerica che serve da indicatore di quello che è avvenuto sulla tastiera, e, dopo l'esecuzione della CALL KEY, contiene uno dei codici che seguono:

Sottoprogramma KEY

- +1 = è stato premuto un nuovo tasto dopo l'ultima esecuzione della routine CALL KEY
- —1 = durante l'esecuzione della CALL KEY è stato premuto lo stesso tasto che era stato premuto durante la precedente esecuzione
- 0 = nessun tasto è stato premuto

Potete controllare questo indicatore di stato nel vostro programma per decidere quale operazione eseguire, come nella linea 110 del programma sulla destra. Questa linea è un controllo che vi dà il tempo di premere un tasto diverso prima che il programma prosegua con l'istruzione successiva.

FIGURA 1. *Suddivisione della tastiera.*
Codici = 0÷19.

CODICI DEI CARATTERI PER LA SUDDIVISIONE DELLA TASTIERA

CODICI	TASTI*	CODICI	TASTI*
0	X,M	10	5,0
1	A,H	11	T,P
2	S,J	12	F,L
3	D,K	13	V,.(punto)
4	W,U	14	C,,(virgola)
5	E,I	15	Z,N
6	R,O	16	B,/(barra obliqua)
7	2,7	17	G; (punto e virgola)
8	3,8	18	Q,Y
9	4,9	19	1,6

* Notate che il primo dei due tasti è sulla sinistra della tastiera ed il secondo è sulla destra.

Esempi:

```
>NEW

>100 CALL KEY(0,KEY,STATUS)
>110 IF STATUS=0 THEN 100
>120 NOTE=KEY-64
>130 ON NOTE GOTO 250,270,150
 ,170,190,210,230
>140 GOTO 100
>150 NOTE=262
>160 GOTO 280
>170 NOTE=294
>180 GOTO 280
>190 NOTE=330
>200 GOTO 280
>210 NOTE=349
>220 GOTO 280
>230 NOTE=392
>240 GOTO 280
>250 NOTE=440
>260 GOTO 280
>270 NOTE=494
>280 CALL SOUND(100,NOTE,2)
>290 GOTO 100
>RUN

--EMETTE UNA NOTA DIVERSA
  DELLA SCALA SE PREMÈTE
  UN TASTO DA "A" A "G"

  PREMETE CLEAR PER FERMARE
  IL PROGRAMMA
```

Sottoprogramma KEY

3	4	7	2	14	12	1	6	15		5
1	2	3	4	5	6	7	8	9	0	=

		11								
Q	W	E	R	T	Y	U	I	O	P	/

	8	9								13
A	S	D	F	G	H	J	K	L	;	ENTER

		10								
SHIFT	Z	X	C	V	B	N	M	,	.	SHIFT

ALPHA LOCK	CTRL	SPACE	FCTN

FIGURA 2. *Tastiera Standard del TI-99/4.*
Unità tastiera = 3. Maiuscole e minuscole
diventano tutte maiuscole.
Codici Funzione = 1÷15.
Nessun carattere di controllo.

131	132	135	130	142	140	129	134	143		133
1	2	3	4	5	6	7	8	9	0	=
							30	31		29

		139								
Q	W	E	R	T	Y	U	I	O	P	/
17	23	5	18	20	25	21	9	15	16	

	136	137								141
A	S	D	F	G	H	J	K	L	;	ENTER
1	19	4	6	7	8	10	11	12	28	

		138								
SHIFT	Z	X	C	V	B	N	M	,	.	SHIFT
	26	24	3	22	2	14	13		27	

ALPHA LOCK	CTRL	SPACE	FCTN

FIGURA 3. *Tastiera Pascal.*
Unità Tastiera = 4. Maiuscole e minuscole attive.
Codici funzione = 129÷143.
Codici dei caratteri di controllo = 1÷31.

3	4	7	2	14	12	1	6	15		5
1	2	3	4	5	6	7	8	9	0	=
							158	159		157

		11								
Q	W	E	R	T	Y	U	I	O	P	/
145	151	133	146	148	153	149	137	143	144	187

	8	9								13
A	S	D	F	G	H	J	K	L	;	ENTER
129	147	132	134	135	136	138	139	140	156	

		10								
SHIFT	Z	X	C	V	B	N	M	,	.	SHIFT
	154	152	131	150	130	142	141	128	155	

ALPHA LOCK	CTRL	SPACE	FCTN

FIGURA 4. *Tastiera BASIC.*
Unità tastiera = 5. Maiuscole e minuscole attive.
Codici funzione = 1÷15.
Codici dei caratteri di controllo =
128÷159, 187

Sottoprogramma KEY

TASTI DI FUNZIONE E DI CONTROLLO

Sono assegnati dei codici anche ai tasti di funzione e di controllo, in modo da poterli usare anche nel sottoprogramma CALL KEY in TI BASIC. I diversi codici assegnati dipendono dall'*unità tastiera* specificata nalla CALL KEY.

Codici dei tasti funzione

Modo BASIC	*Modo Pascal*	*Nome della Funzione*	*Tasti di Funzione*
1	129	**AID** (aiutare)	**FCTN 7**
2	130	**CLEAR** (annullare)	**FCTN 4**
3	131	**DEL**ete (cancellare)	**FCTN 1**
4	132	**INS**ert (inserire)	**FCTN 2**
5	133	**QUIT** (terminare)	**FCTN =**
6	134	**REDO** (rifare)	**FCTN 8**
7	135	**ERASE** (eliminare)	**FCTN 3**
8	136	**LEFT** arrow (a sinistra)	**FCTN S**
9	137	**RIGHT** arrow (a destra)	**FCTN D**
10	138	**DOWN** arrow (in giù)	**FCTN X**
11	139	**UP** arrow (in su)	**FCTN E**
12	140	**PROC'D** (continuare)	**FCTN 6**
13	141	**ENTER** (avanti)	**ENTER**
14	142	**BEGIN** (iniziare)	**FCTN 5**
15	143	**BACK** (indietro)	**FCTN 9**

Codici dei tasti di controllo

Codici Modo BASIC	*Codici Modo Pascal*	*Codici Mnemonici*	*Tasto premuto*	*Commenti*
129	1	SOH	CONTROL A	Inizio intestazione
130	2	STX	CONTROL B	Inizio testo
131	3	ETX	CONTROL C	Fine testo
132	4	EOT	CONTROL D	Fine trasmissione
133	5	ENQ	CONTROL E	Richiesta
134	6	ACK	CONTROL F	Riconoscimento
135	7	BEL	CONTROL G	Campanello
136	8	BS	CONTROL H	Ritorno unitario
137	9	HT	CONTROL I	Tabulazione orizzontale
138	10	LF	CONTROL J	Avanzamento di una linea
139	11	VT	CONTROL K	Tabulazione verticale
140	12	FF	CONTROL L	Avanzamento foglio
141	13	CR	CONTROL M	Ritorno carrello
142	14	SO	CONTROL N	Scorrimento in fuori
143	15	SI	CONTROL O	Scorrimento in dentro
144	16	DLE	CONTROL P	Data link escape
145	17	DC1	CONTROL Q	Controllo dispositivo 1(X-ON)
146	18	DC2	CONTROL R	Controllo dispositivo 2
147	19	DC3	CONTROL S	Controllo dispositivo 3(X-OFF)
148	20	DC4	CONTROL T	Controllo dispositivo 4
149	21	NAK	CONTROL U	Mancato riconoscimento
150	22	SYN	CONTROL V	Carattere sincronizzazione
151	23	ETB	CONTROL W	Fine trasmissione blocco
152	24	CAN	CONTROL X	Cancellazione
153	25	EM	CONTROL Y	Fine del grassetto
154	26	SUB	CONTROL Z	Sostituzione
155	27	ESC	CONTROL .	Uscita
156	28	FS	CONTROL ;	Separatore file
157	29	GS	CONTROL =	Separatore gruppi
158	30	RS	CONTROL 8	Separatore record
159	31	US	CONTROL 9	Separatore unità

Vedere la sezione "ASC-Valore ASCII", se si vuole la lista completa dei caratteri standard ASCII.

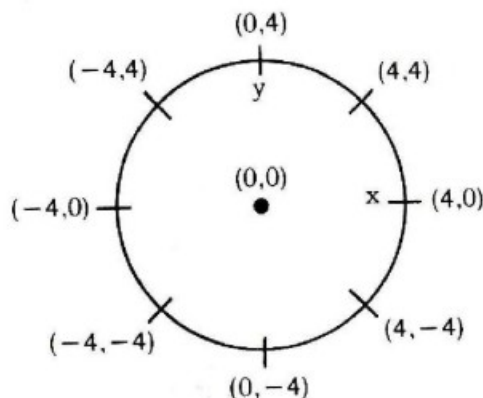
Sottoprogramma JOYST

CALL JOYST *(unità tastiera, x, y)*

Il sottoprogramma JOYST vi consente di inviare al calcolatore informazioni sulla posizione del joystick dei Controlli a distanza (disponibili a parte).

L'*unità Tastiera* è un'espressione numerica con valore da 1 a 4.

- 1 = controllo 1
- 2 = controllo 2
- 3, 4, 5 = modi particolari per la tastiera della console

Le *unità* 3, 4 e 5 ridefiniscono la tastiera della console in un particolare modo di operazioni, come è spiegato nel paragrafo "KEY". In questo caso ingressi da controlli a distanza non vengono riconosciuti correttamente.
x e *y* devono essere variabili numeriche, a cui il sottoprogramma assegna un valore intero pari a -4, +4 o 0 a seconda della posizione del joystick, come si vede nella figura seguente. Il primo valore tra parentesi è *x*, e il secondo è *y*.

Potete usare questi valori nel programma riferendovi alle variabili che li contengono.

Troverete informazioni più dettagliate nel manuale dei "Controlli a distanza."

Esempi:

```
>NEW

>100 CALL CLEAR
>110 CALL CHAR(42,"FFFFFFFFFF
 FFFFFF")
>120 PRINT "SCELTA COLORI"
>125 INPUT "SCHERMO?":V
>130 INPUT "BLOCCO?":B
>140 CALL CLEAR
>150 CALL SCREEN(V)
>160 CALL COLOR(2,B,1)
>170 CALL JOYST(2,X,Y)
>180 A=X*2.2+16.6
>190 C=Y*1.6+12.2
>200 CALL HCHAR(C,A,42)
>210 GOTO 170
>RUN

--LO SCHERMO SI PULISCE

 SCELTA COLORI
 SCHERMO?14
 BLOCCO?9

--LO SCHERMO SI PULISCE

--IL BLOCCO COLORATO SI
  MUOVE IN CORRISPONDENZA
  AI MOVIMENTI DEL JOYSTICK

  PREMERE CLEAR PER FERMARE

  IL PROGRAMMA
```

Funzioni Aritmetiche

Introduzione

Il TI BASIC vi mette a disposizione molte funzioni per operazioni speciali, che sono descritte in questo paragrafo.

L'Extended BASIC* ha una più vasta gamma di funzioni numeriche, come MAX,MIN,PI etc, che risparmiano all'utente gran parte del lavoro di programmazione, il più noioso.

* in vendita separatamente

Esempi:

ABS - Valore assoluto

ABS *(espressione numerica)*

Restituisce il valore assoluto dell'argomento, cioè il valore ottenuto calcolando l'*espressione numerica*. Questa espressione viene calcolata seguendo le regole che già conoscete (vedi "Espressioni Numeriche"). Se l'argomento è positivo, la funzione "valore assoluto" restituisce l'argomento stesso. Se è negativo restituisce il negativo dell'argomento. Perciò, dato l'argomento X

- Se $X \geq 0$, ABS(X) = X
- Se $X \geq 0$, ABS(X) = X
  (esempio: ABS (-3) = -(-3) = 3)

```
>NEW

>100 A=-27.36
>110 B=9.7
>120 PRINT ABS(A);ABS(B)
>130 PRINT ABS(3.8);ABS(-4.5)
>140 PRINT ABS(-3*2)
>150 PRINT ABS(A*(B-3.2))
>160 END
>RUN
  27.36  9.7
  3.8  4.5
  6
  177.84

  ** DONE **
```

ATN -Arcotangente

ATN *(espressione numerica)*

La funzione arcotangente vi restituisce l'arcotangente dell'argomento, cioè del valore che si ottiene calcolando l'*espressione numerica*. L'espressione viene calcolata con le normali regole che già conosciamo. ATN(x) vi dà l'angolo in radianti la cui tangente è x. Se volete l'angolo equivalente in gradi, dovete moltiplicarlo per (180/4*ATN(1)) o 57.295779513079, cioè 180/π. Il valore ottenuto dalla funzione arcotangente è sempre compreso tra $-\pi/2 < \text{ATN}(x) < \pi/2$.

Esempi:

```
>NEW
>100 PRINT ATN(.44)
>110 PRINT ATN(1E127)
>120 PRINT ATN(1E-129);ATN
 (0)
>130 PRINT ATN(.3)*57.2957
 79513079
>140 PRINT ATN(.3)*(180/(4
 *ATN(1)))
>150 END
>RUN
  .4145068746
  1.570796327
  0  0
  16.69924423
  16.69924423

  ** DONE **
```

COS -Coseno

COS *(espressione numerica)*

La funzione coseno vi dà il coseno dell'argomento x, dove x è un angolo in radianti, ottenuto calcolando l'*espressione numerica*, con le solite regole. Se l'angolo è in gradi, dovete moltiplicarlo per π/180 per ottenere l'equivalente in radianti. Per ottenere π/180 potete fare (4*ATN(1))/180 o 0.01745329251994. Se date un valore di x con $|x| \geqslant 1.5707963266375 \cdot 10^{10}$ compare il messaggio "BAD ARGUMENT", e il programma si ferma.

```
>NEW

>100 A=1.047197551196
>110 B=60
>120 C=.01745329251994
>130 PRINT COS(A);COS(B*C)

>140 PRINT COS(B*(4*ATN(1)
)/180)
>150 END
>RUN
  .5  .5
  .5

  ** DONE **

 >PRINT COS(2.2E11)

  * BAD ARGUMENT
```

EXP - Esponenziale

EXP *(espressione numerica)*

La funzione esponenziale vi restituisce il valore di e^x, dove e=2.718281828 e x è il valore ottenuto calcolando l'*espressione numerica*, con le solite regole. La funzione esponenziale e l'inverso del logaritmo naturale (LOG), cioè X=EXP(LOG(X)).

```
>NEW
>100 A=3.79
>110 PRINT EXP(A);EXP(9)
>120 PRINT EXP(A*2)
>130 PRINT EXP(LOG(2))
>140 END
>RUN
  44.25640028  8103.083928
  1958.628965
  2
  ** DONE **
```

INT - Intero

INT *(espressione numerica)*

Questa funzione vi dà il numero intero più grande, inferiore o uguale all'argomento cioè al valore ottenuto calcolando l'*espressione numerica* con le regole che già conoscete. Il valore restituito è l'intero più vicino a sinistra dell'argomento. Quindi, nei numeri positivi viene troncata la parte decimale,per quelli negativi si ottiene l'intero immediatamente successivo (esempio INT(-2.3)= —3). Se l'argomento è intero, la funzione restituisce l'argomento stesso.

Esempi:

```
>NEW

>100 B=.678
>110 A=INT(B*100+.5)/100
>120 PRINT A;INT(B)
>130 PRINT INT(-2.3);INT(2.2)
>140 STOP
>RUN
  .68  0
 -3  2

 ** DONE **
```

LOG - Logaritmo Naturale

Vi dà il logaritmo naturale dell'argomento, cioè del valore ottenuto calcolando l'*espressione numerica* con le solite regole. Il logaritmo naturale di x si indica in genere come $\log_e (x)$. La funzione logaritmo è l'inverso dell'esponenziale, cioè X=LOG(EXP(X)).

```
>NEW

>100 A=3.5
>110 PRINT LOG(A);LOG(A*2)
>120 PRINT LOG(EXP(2))
>130 STOP
>RUN
  1.252762968  1.945910149
  2.

 ** DONE **
```

L'argomento della funzione logaritmo naturale deve essere maggiore di zero. Se date un valore minore o uguale a zero, compare il messaggio "BAD ARGUMENT", e il programma si ferma.

```
 PRINT LOG(-3)

 * BAD ARGUMENT
```

Per trovare il logaritmo in un'altra base B, usate la seguente formula:

$\log_B (x) = \log_e (x)/\log_e (B)$

Per esempio, $\log_{10} (3) = \log_e (3)/\log_e (10)$

```
>PRINT LOG(3)/LOG(10)
 .4771212547
```

RANDOMIZE

RANDOMIZE [*origine*]

L'istruzione RANDOMIZE si usa con la funzione di generazione di numeri casuali (RND). Se non si usa la RANDOMIZE, la RND genera la stessa sequenza di numeri pseudo-casuali ogni volta che viene eseguita. Se si usa la RANDOMIZE senza *origine*, la RND genera una sequenza diversa e imprevedibile di numeri casuali ogni volta che viene eseguita. Se si usa la RANDOMIZE specificando un'*origine*, la sequenza di numeri casuali generati dalla RND dipende dal valore dell'*origine*. Con la stessa *origine*, si genera sempre la stessa sequenza, con *origini* diverse si generano sequenze diverse. L'*origine* può essere un' espressione numerica qualsiasi. Il numero usato effettivamente come *origine* è costituito dai primi due byte della sua rappresentazione interna. Quindi può capitare che si generi la stessa sequenza di numeri anche se date differenti *origini*. Per esempio, RANDOMIZE 1000 e RANDOMIZE 1099 danno gli stessi due primi byte nella rappresentazione interna, e quindi la stessa sequenza di numeri. Se l'*origine* specificata non è un numero intero, viene usato il valore INT*(origine)* (vedi la funzione "INT -Intero").

Esempi:

```
>NEW

>100 RANDOMIZE 23
>110 FOR I=1 TO 5
>120 PRINT INT(10*RND)+1
>130 NEXT I
>140 STOP
>RUN
  6
  4
  3
  8
  8

 ** DONE **
```

RND - Numeri Casuali

RND

La funzione di generazione di numeri casuali dà il numero pseudo-casuale successivo di una certa sequenza, maggiore o uguale a 0 e minore di 1. La sequenza è sempre la stessa ogni volta che il programma viene eseguito, a meno che nel programma ci sia l'istruzione RANDOMIZE.

```
>NEW

>100 FOR I=1 TO 5
>110 PRINT INT(10*RND)+1
>120 NEXT I
>130 END
>RUN
  6
  4
  6
  4
  3

 ** DONE **
```

Per ottenere numeri casuali interi compresi tra A e B (A>B), usate la formula

INT((B-A+1)*RND)+A

```
>NEW

>100 REM INTERI RANDOM TRA
 1 E 20 ESTREMI INCLUSI
>110 FOR I=1 TO 5
>120 C=INT(20*RND)+1
>130 PRINT C
>140 NEXT I
>150 END
>RUN
  11
  8
  11
  8
  6

 ** DONE **
```

SGN - Segno

SGN *(espressione numerica)*

La funzione "segno" vi dà il segno algebrico dell'argomento, cioè del valore ottenuto calcolando l'*espressione numerica* con le solite regole. Il valore restituito è diverso a seconda del valore dell'argomento, e precisamente, per un argomento X:

- X<0, SGN(X) = —1
- X=0, SGN(X) = 0
- X>0, SGN(X) = 1

Esempi:

```
>NEW

>100 A=-23.7
>110 B=6
>120 PRINT SGN(A);SGN(0);SGN(B)
>130 PRINT SGN(-3*3);SGN(B*2)
>140 END
>RUN
 -1  0  1
 -1  1

 ** DONE **
```

SIN - Seno

SIN *(espressione numerica)*

La funzione seno vi dà il seno dell'argomento x, dove x è un angolo in radianti. L'argomento è il valore ottenuto calcolando con le regole abituali l'*espressione numerica*. Se l'angolo è in gradi, per trasformarlo in radianti dovete moltiplicarlo per $\pi/180$, cioè (4*ATN(1))/180 o 0.01745329251944. Notate che se date un valore di x con $|x| \geqslant 1.5707963266375*10^{10}$, compare il messaggio "BAD ARGUMENT" e il programma si ferma.

```
>NEW

>100 A=.5235987755982
>110 B=30
>120 C=.01745329251994
>130 PRINT SIN(A);SIN(B*C)
>140 PRINT SIN(B*(4*ATN(1))/180)
>150 END
>RUN
  .5  .5
  .5

 ** DONE **


>PRINT SIN(1.9E12)

 * BAD ARGUMENT
```

SQR - Radice Quadrata

SQR*(espressione numerica)*

Questa funzione vi dà la radice quadrata positiva dell'argomento, cioè del valore che si ottiene calcolando l'*espressione numerica* con le solite regole che già conoscete. SQR(x) è equivalente a x ∧ (1/2). L'argomento non può essere negativo, altrimenti compare il messaggio "BAD ARGUMENT" e il programma si ferma.

Esempi:

```
>NEW

>100 PRINT SQR(4);4^(1/2)
>110 PRINT SQR(10)
>130 END
>RUN
  2  2
  3.16227766

 ** DONE **

>PRINT SQR(-5)

 * BAD ARGUMENT
```

TAN - Tangente

TAN*(espressione numerica)*

Questa funzione vi dà la tangente dell'argomento x, dove x è un angolo in radianti. L'argomento è il valore che si ottiene calcolando l'*espressione numerica* con le solite regole. Se l'angolo è in gradi, per averlo in radianti dovete moltiplicarlo per $\pi/180$, cioè (4*ATN(1))/180 o 0.017453251994. Se date un valore di x con $|x| \geqslant 1.5707963266375*10^{10}$, compare il messaggio "BAD ARGUMENT" e il programma si ferma.

```
>NEW

>100 A=.7853981633973
>110 B=45
>120 C=.01745329251994
>130 PRINT TAN(A);TAN(B*C)
>140 PRINT TAN(B*(4*ATN(1))/180)
>150 END
>RUN
  1.   1.
  1

 ** DONE **

>PRINT TAN(1.76E10)

 * BAD ARGUMENT
```

Funzioni di Stringa

Introduzione

Il TI BASIC vi mette a disposizione molte altre funzioni oltre a quelle numeriche. In questo paragrafo vi presentiamo le funzioni di stringa. Tra queste, alcune usano una stringa in diversi modi per produrre un risultato numerico, o ancora una stringa. Usando il vostro calcolatore troverete molti modi di usare queste funzioni. Potete anche definire delle vostre particolari funzioni di stringa (vedi il paragrafo "Funzioni definite dall'utente"). Notate che una funzione di stringa il cui nome finisce con il $ (ad esempio CHAR$) dà un risultato di stringa, e non può essere usata in espressioni numeriche.

Esempi:

ASC - Valore ASCII

ASC*(espressione di stringa)*

Questa funzione vi dà il codice ASCII del primo carattere della stringa specificata nell'*espressione di stringa*. Nella pagina seguente c'è una tabella con tutti i caratteri ASCII e i loro codici.

```
>NEW

>100 A$="CIAO"
>110 C$="GIANNI ROSSI"
>120 C=ASC(C$)
>130 B$="IL VALORE ASCII DI "
>140 PRINT B$;"C E` ";ASC(A$)
>150 PRINT B$;"G E` ";C
>160 PRINT B$;"N E` ";ASC("NOME"
 )
>170 PRINT B$;"1 E` ";ASC("1")
>180 PRINT CHR$(ASC(A$))
>190 END
>RUN
  IL VALORE ASCII DI C E` 67
  IL VALORE ASCII DI G E` 71
  IL VALORE ASCII DI N E` 78
  IL VALORE ASCII DI 1 E` 49
  C

** DONE **
```

ASC - Valore ASCII

CODICI DEI CARATTERI ASCII

I caratteri definiti per l'Home Computer TI-99/4A sono quelli standard ASCII con codici da 32 a 127. Nella seguente tabella sono elencati questi cartteri e i codici corrispondenti.

CODICE ASCII	CARATTERE	CODICE ASCII	CARATTERE	CODICE ASCII	CARATTERE
32	(spazio)	65	A	97	A
33	! (punto esclamativo)	66	B	98	B
34	" (virgolette)	67	C	99	C
35	# (numero o libbra)	68	D	100	D
36	$ (dollaro)	69	E	101	E
37	% (per cento)	70	F	102	F
38	& ("e" commerciale)	71	G	103	G
39	' (apostrofo)	72	H	104	H
40	((parentesi aperta)	73	I	105	I
41	) (parentesi chiusa)	74	J	106	J
42	* (asterisco)	75	K	107	K
43	+ (segno più)	76	L	108	L
44	, (virgola)	77	M	109	M
45	- (segno meno)	78	N	110	N
46	. (punto)	79	O	111	O
47	/ (barra obliqua)	80	P	112	P
48	0	81	Q	113	Q
49	1	82	R	114	R
50	2	83	S	115	S
51	3	84	T	116	T
52	4	85	U	117	U
53	5	86	V	118	V
54	6	87	W	119	W
55	7	88	X	120	X
56	8	89	Y	121	Y
57	9	90	Z	122	Z
58	: (due punti)	91	[(parentesi quadra aperta)	123	{ (parentesi graffa aperta)
59	; (punto e virgola)	92	\ (barra obliqua inversa)	124	
60	< (minore)	93	] (parentesi quadra chiusa)	125	} (parentesi graffa chiusa)
61	= (uguale)	94	∧ (elevamento a potenza)	126	— (tilde)
62	> (maggiore)	95	- (trattino)	127	DEL (visualizzato come spazio)
63	? (punto interrogativo)	96	` (accento grave)		
64	@ (segno at)				

Questi caratteri sono raggruppati in dodici *insiemi* per l'uso nei programmi di grafica a colori.

Insieme	*Codici Caratteri*	*Insieme*	*Codici Caratteri*	*Insieme*	*Codici Caratteri*
1	32-39	5	64-71	9	96-103
2	40-47	6	72-79	10	104-111
3	48-55	7	80-87	11	112-119
4	56-63	8	88-95	12	120-127

Esistono infine altri due caratteri del TI-99/4A, il *cursore*, con codice 30, e il *carattere di margine*, con codice 31.

CHR$ - Carattere

CHR$ *(espressione numerica)*

Questa funzione vi restituisce il carattere corrispondente al codice ASCII specificato nell'argomento. L'argomento è il valore che si ottiene calcolando con le solite regole l'*espressione numerica*. Se questo valore non è intero, viene arrotondato. Se il valore dell'argomento è compreso tra 32 e 127, viene prodotto un carattere standard. Se è tra 128 e 159 compresi, e per quel valore è stato definito un carattere grafico speciale, si ottiene questo. Se l'argomento non corrisponde a un carattere definito (cioè un carattere standard o un carattere grafico predefinito), si ottiene il carattere corrispondente a quello che c'è in memoria in quel momento.

Esempi:

```
>NEW

>100 A$=CHR$(72)&CHR$(73)&CHR$(
 3)
>110 PRINT A$
>120 CALL CHAR(97,"0103070F1F3F
 FFF")
>130 PRINT CHR$(32);CHR$(97)
>140 PRINT CHR$(3*14)
>150 PRINT CHR$(ASC("+"))
>160 END
>RUN
 HI!
  ◢
  *
  +
 ** DONE **
```

Se l'argomento ha un valore minore di zero o maggiore di 32767, compare il messaggio "BAD VALUE" e il programma si ferma.

```
>PRINT CHR$(33010)

 * BAD VALUE
```

LEN - Lunghezza

LEN*(espressione di stringa)*

Vi dà il numero di caratteri della stringa specificata nell'argomento. L'argomento è la stringa che si ottiene calcolando l'*espressione di stringa*, con le regole note. Una stringa nulla ha lunghezza zero. Ricordate che lo spazio viene contato come un carattere.

```
>NEW

>100 NOME$="SONIA"
>110 CITTA$="MILANO"
>120 MSG$="CIAO"&"BELLO"
>130 PRINT NOME$;LEN(NOME$)
>140 PRINT CITTA$;LEN(CITTA$)
>150 PRINT MSG$;LEN(MSG$)
>160 PRINT LEN(NOME$&CITTA$)
>170 PRINT LEN("HI!")
>180 STOP
>RUN
SONIA 5
MILANO 6
CIAO BELLO 10
 11
 3

 ** DONE **
```

POS - Posizione

POS(*stringa-1, stringa-2, espressione numerica*)

Vi dà la posizione in cui si incontra per la prima volta la *stringa-2* nella *stringa-1*. Sia *stringa-1* che *stringa-2* sono espressioni di stringa. L'espressione numerica viene calcolata e, se il caso, arrotondata a un intero *n*. Si usano le solite regole sia per il calcolo delle espressioni di stringa che per quelle numeriche. La ricerca della *stringa-2* inizia dall'*n-esimo* carattere della *stringa-1*. Se viene trovata, la funzione restituisce la posizione del primo carattere della *stringa-2* nella *stringa-1*, altrimenti restituisce zero. La posizione del primo carattere della *stringa-1* corrisponde al valore 1. Se *n* è maggiore del numero di caratteri della *stringa-1*, si ottiene zero. Se invece *n* è minore di zero, compare il messaggio "BAD VALUE", e il programma si ferma.

```
>NEW
>100 MSG$="CIAO BELLO! COME STAI
?"
>110 PRINT "C";POS(MSG$,"C",1)
>120 C$="LO"
>130 PRINT C$;POS(MSG$,C$,1),"OM
";POS(MSG$,"OM",12)
>140 PRINT "HI";POS(MSG$,"HI",1)
>150 END
>RUN
 C 1
 LO  9       OM  14
 HI  0

 ** DONE **
```

SEG$ - Segmento di Stringa

SEG(*espressione di stringa, espressione numerica 1, espressione numerica 2*)

Questa funzione vi dà una parte della stringa indicata dall'*espressione di stringa*. L'*espressione numerica 1* identifica la posizione del carattere della stringa originale che è il primo carattere della parte voluta. Il primo carattere della stringa originale è nella posizione 1. La parte di stringa ottenuta ha lunghezza specificata dall'*espressione numerica 2*. Anche qui si usano le solite regole per il calcolo delle espressioni di stringa e per quelle numeriche.

```
>NEW

>100 MSG$="CIAO BELLO! COME STAI
?"
>110 REM LA SUBSTRING INIZIA IN
POSIZIONE 13 ED E' LUNGA 10
>120 PRINT SEG$(MSG$,13,10)
>130 END
>RUN
 COME STAI?

 ** DONE **
```

Nella spiegazione che segue, A$ è l'*espressione di stringa*, X è l'*espressione numerica 1* e Y l'*espressione numerica 2*. Se date un valore di X maggiore della lunghezza di A$ (linea 110), o un valore di Y uguale a zero (linea 120), ottenete una stringa nulla. Se date un valore di Y maggiore della parte di A $ che inizia alla posizione specificata da X (linea 130), otterrete ugualmente questa parte di A$.

```
>NEW

>100 MSG$="IO SONO UN CALCOLATOR
E"
>110 PRINT SEG$(MSG$,24,1)
>120 PRINT SEG$(MSG$,11,0)
>130 PRINT SEG$(MSG$,8,20)
>140 END
>RUN
 O

 UN CALCOLATORE.

 ** DONE **
```

Se date un valore di X minore o uguale a zero, e/o un valore di Y minore di zero, compare il messaggio "BAD VALUE" e il programma si ferma.

```
>PRINT SEG$(MSG$,-1,10)

 * BAD VALUE
```

STR$ - Numero - Stringa

STR$ *(espressione numerica)*

Questa funzione converte in stringa il numero specificato dall'argomento. L'argomento è il valore che si ottiene calcolando l'*espressione numerica* con le solite regole. La stringa che si ottiene è una corretta rappresentazione di una costante numerica, senza spazi prima e dopo. Ad esempio, se B=69.5, allora STR(B) è la stringa "69.5". Naturalmente, sulle stringhe che si ottengono si possono eseguire solo operazioni di stringa. La funzione numero-stringa è l'inverso della funzione VAL, spiegata qui di seguito. Notate nell'esempio che i numeri convertiti in stringa non hanno spazi nè prima nè dopo.

Esempi:

```
>NEW

>100 A=-26.3
>110 PRINT STR$(A);" ";A
>120 PRINT 15.7;STR$(15.7)
>130 PRINT STR$(VAL("34.8"))
>140 END
>RUN
 -26.3 -26.3
  15.7  15.7
 34.8
 -26.3 -26.3
  15.7 15.7
 34.8


 ** DONE **
```

VAL - Valore

VAL*(espressione di stringa)*

La funzione "valore" è l'inverso della "numero-stringa" STR$ che abbiamo già visto. Se la stringa specificata nell'*espressione di stringa* è una rappresentazione corretta di una costante numerica, viene convertita nella costante numerica stessa. Per esempio, se A$ = "1234", VAL(A$)=1234. L'espressione di stringa viene calcolata con le solite regole. Se la stringa non è una rappresentazione corretta di un numero, se ha lunghezza zero, o maggiore di 254 caratteri, compare il messaggio "BAD ARGUMENT" e il programma si ferma.

```
>NEW

>100 P$="23.6"
>110 N$="-4.7"
>120 PRINT VAL(P$);VAL(N$)
>130 PRINT VAL("52"&".5")
>140 PRINT VAL(N$&"E"&"12")
>150 PRINT STR$(VAL(P$))
>160 END
>RUN
  23.6 -4.7
  52.5
 -4.7E+12
 23.6

 ** DONE **
```

Funzioni Definite dall'Utente

Introduzione

Oltre alle funzioni di TI BASIC che abbiamo visto nei paragrafi precedenti, l'utente può usare funzioni da lui stesso definite, che semplificano la programmazione evitando la scrittura ripetuta di espressioni complesse. Una funzione definita con l'istruzione DEF può essere usata in qualsiasi punto del programma semplicemente richiamandola con il nome che le è stato assegnato.

DEF

DEF {*nome di funzione numerica* [(*parametro*)]=*espressione numerica*
nome di funzione di stringa [(*parametro*)]=*espressione di stringa*}

L'istruzione DEF vi consente di definire funzioni particolari per usarle in un programma. Il *nome della funzione* può essere qualsiasi nome di variabile. Il *parametro*, se specificato, deve essere tra parentesi e può avere un qualsiasi nome di variabile. Se l'espressione specificata dà un valore di stringa, il *nome della funzione* deve essere una variabile di stringa (cioè deve avere come ultimo carattere il $).

Esempi:

```
>NEW

>100 DEF PI=4*ATN(1)
>110 PRINT COS(60*PI/180)
>120 END
>RUN
  .5

 ** DONE **
```

L'espressione DEF definisce la funzione in base al *parametro*(se specificato), a variabili, costanti e altre funzioni. Una volta definita, potete usarla in qualsiasi espressione numerica o di stringa richiamandola con il suo *nome*. Se nella DEF è stato specificato un *parametro*, il *nome della funzione*, quando la si richiama, deve essere seguito da un argomento tra parentesi. Se il *parametro* non è stato specificato, ogni volta che la funzione viene richiamata in un'espressione,viene calcolata in base ai valori attuali delle variabili che compaiono nella DEF.

```
>NEW

>100 REM CALCOLO Y=X*(X-3)
>110 DEF Y=X*(X-3)
>120 PRINT " X  Y"
>130 FOR X=-2 TO 5
>140 PRINT X;Y
>150 NEXT X
>160 END
>RUN
  X  Y
 -2  10
 -1  4
  0  0
  1 -2
  2 -2
  3  0
  4  4
  5  10

 ** DONE **
```

DEF

Se specificate un *parametro*, quando si incontra la funzione in un'espressione, viene calcolato l'argomento e il suo valore assegnato al *parametro*. Dopodichè viene calcolata la funzione in base al nuovo valore del *parametro* e ai valori attuali delle altre variabili che compaiono nella DEF.

Esempi:

```
>NEW

>100 REM STAMPA NOME ROVESCIATO
>110 DEF BACK$(X)=SEG$(NOME$,X,1
)
>120 INPUT "NOME? ":NOME$
>130 FOR I=LEN(NOME$) TO 1 STEP
 -1
>140 RNOME$=RNOME$&BACK$(I)
>150 NEXT I
>160 PRINT NOME$:RNOME$
>170 END
>RUN
 NOME? ROBOT
 ROBOT
 TOBOR

 ** DONE **
```

Il *parametro* usato nella DEF è "locale" per questa istruzione. Ciò significa che non ha niente a che vedere con un'altra eventuale variabile con lo stesso nome usata in un'altra istruzione. Perciò, quando si calcola la funzione, non si influenza il valore di quest'altra variabile.

```
>NEW

>100 DEF FUNC(A)=A*(A+B-5)
>110 A=6.9
>120 B=13
>130 PRINT "B= ";B:"FUNC(3)=";F
 UNC(3):"A= ";A
>140 END
>RUN
 B=  13
 FUNC(3)=  33
 A=  6.9

 ** DONE **
```

Un'istruzione DEF viene eseguita solo quando la funzione che definisce viene richiamata in un'espressione. Quando il calcolatore incontra una DEF nel programma, non esegue alcuna operazione e prosegue all'istruzione successiva. Un'istruzione DEF può trovarsi in qualsiasi punto di un programma, e non deve necessariamente precedere dal punto di vista logico un riferimento alla funzione che definisce. Deve però avere un numero di linea più basso di qualsiasi istruzione in cui la funzione viene richiamata.

```
>NEW

>100 REM CALCOLO APPROSSIMATO DI
 F'(X)
>110 INPUT "X=? ":X
>120 IF ABS(X)>.01 THEN 150
>130 H=.00001
>140 GOTO 180
>150 H=.001*ABS(X)
>160 DEF F(Z)=3*Z^3-2*Z+1
>170 DEF DER(X)=(F(X+H)-F(X-H))/
 (2*H)
>180 PRINT "F'(";STR$(X);")=";DE
 R(X)
>190 END
>RUN
X=? .1
 F'(.1)= -1.90999997

 ** DONE **
```

DEF

Nella DEF, la funzione che viene definita non può richiamare se stessa, ne direttamente (DEF B=B*2), ne indirettamente (DEF F=G; DEF G=F). Il *parametro* non può essere usato come matrice, mentre potete usare un elemento di una matrice nella definizione della funzione, se la matrice non ha lo stesso nome del *parametro*.

Esempi:

```
>NEW

>100 DEF GX(X)=GX(2)*X
>110 PRINT GX(2)
>120 END
>RUN

 * MEMORY FULL IN 110

>100 DEF GX(A)=A(3)^2
>RUN


* NAME CONFLICT IN 100
```

Se specificate un *parametro* nel definire la funzione, quando la richiamate dovete specificare un argomento. In caso contrario, non potete farlo.

```
>NEW

>100 DEF SQUARE(X)=X*X
>110 PRINT SQUARE
>120 END
>RUN

 * NAME CONFLICT IN 110

>100 DEF PI=3.1416
>110 PRINT PI(2)
>RUN

* NAME CONFLICT IN 110
```

Matrici

Introduzione

Una matrice è un insieme di variabili organizzate in modo da poter essere usate facilmente in un programma. Il tipo di organizzazione più comune è la "lista", cioè una matrice a una dimensione. Ogni variabile della lista si chiama "elemento" della matrice. La lunghezza della lista è limitata solo dalla quantità di memoria disponibile.

Con le possibilità di manipolazione delle matrici del TI BASIC potete fare diversi tipi di operazioni su di una lista - stampare gli elementi dal primo all'ultimo o viceversa, riordinarli, sommarli fra di loro, moltiplicarli, e selezionarne alcuni per particolari elaborazioni.

In TI BASIC una matrice può iniziare dall'elemento 1 o 0. Con l'istruzione OPTION BASE potete controllare quale di queste due convenzioni usa il calcolatore. Noi supporremo d'ora in poi che il primo elemento sia l'elemento 1.

Potete creare due liste di quattro numeri ognuna, e stampare tutte le possibili combinazioni dei numeri di entrambe. Chiamiamo X la prima matrice, e Y la seconda. Poichè X e Y rappresentano *insiemi* di numeri, e non variabili singole, il calcolatore deve avere modo di riferirsi ad ogni elemento della matrice. Per far ciò, dovete fornire un puntatore, detto "indice", che individua all'interno della matrice il particolare elemento che il calcolatore deve usare. L'indice è tra parentesi, e segue immediatamente il nome della matrice. Può essere esplicito, come X(3), che indica il terzo elemento della matrice X, o può essere una variabile, come per X(T), dove il valore di T individua la posizione dell'elemento nella matrice. In ogni caso deve essere intero maggiore o uguale a zero.

Il programma sulla destra accoppia i numeri delle due matrici X e Y. Notate che con le tecniche di organizzazione in matrici, bastano poche righe di programma per una procedura relativamente complessa.

Esempi:

```
>NEW

>100 REM ACCOPPIAMENTO LISTE
>110 REM DA 120 A 150 SI ASSEGNA
NO VALORI ALLA LISTA X
>120 FOR T=1 TO 4
>130 READ X(T)
>140 NEXT T
>10 DATA 1,3,5,7
>160 REM DA 170 A 200 SI ASSEGNA
 NO VALORI ALLA LISTA Y
>170 FOR S=1 TO 4
>180 READ Y(S)
>190 NEXT S
>200 DATA 2,4,6,8
>210 REM DA 220 A 270 SI ACCOPPI
 ANO LE LISTE E SI STAMPA LA COMB
 INAZIONE
>220 FOR T=1 TO 4
>230 FOR S=1 TO 4
>240 PRINT X(T);Y(S);"  ";
>250 NEXT S
>260 PRINT
>270 NEXT T
>280 END
>RUN
  1  2   1  4   1  6   1  8
  3  2   3  4   3  6   3  8
  5  2   5  4   5  6   5  8
  7  2   7  4   7  6   7  8

 ** DONE **
```

DIMension

DIM {*nome matrice(intero 1*[, *intero 2*] [, *intero 3*])}, ...

L'istruzione DIM riserva in memoria lo spazio necessario per matrici numeriche e di stringa. Una matrice può essere dimensionata esplicitamente una sola volta in un programma, e l'istruzione DIM relativa deve trovarsi prima di qualsiasi riferimento alla matrice stessa. Se in una sola DIM dimensionate più di una matrice, i loro nomi devono essere separati da virgole. Il *nome matrice* può essere qualsiasi nome valido di variabile.

In TI BASIC potete usare matrici a una, due o tre dimensioni. Il numero di valori tra parentesi che seguono il nome della matrice è uguale al numero delle dimensioni.

Una matrice mono-dimensionale ha un solo valore intero dopo il suo nome. Le matrici bi-dimensionali sono descritte con due valori interi che definiscono il numero di righe e di colonne. Le matrici tri-dimensionali hanno il nome seguito da tre valori interi che definiscono le loro caratteristiche.

- DIM A(6) - descrive una matrice mono-dimensionale
- DIM A(12,3) - descrive una matrice bi-dimensionale
- DIM A(5,2,11) - descrive una matrice tri-dimensionale

Se una matrice non è dimensionata in un'istruzione DIM, il calcolatore le assegna automaticamente il valore 10 per *intero 1* e, se necessario, per *intero 2 e intero 3*.

Lo spazio per le vostre matrici viene riservato dopo che date il comando RUN, ma prima di iniziare l'esecuzione del programma. Ogni elemento di una matrice di stringa viene considerato di lunghezza zero fin quando non gli assegnate un valore. Se la memoria disponibile non è sufficiente per contenere una matrice con le dimensioni che avete specificato, compare il messaggio "MEMORY FULL" e il programma non viene eseguito.

Esempi:

```
>DIM A(12),B(5)
```

```
>NEW

>100 DIM X(15)
>110 FOR I=1 TO 15
>120 READ X(I)
>130 NEXT I
>140 REM CICLO STAMPA
>150 FOR I=15 TO 1 STEP -1
>160 PRINT X(I);
>170 NEXT I
>180 DATA 1,2,3,4,5,6,7,8,9,10,1
 1,12,13,14,15
>190 END
>RUN
  15  14  13  12  11  10  9
  8  7  6  5  4  3  2  1

 ** DONE **
```

DIMension

Indici di una matrice

Tutte le volte che vi riferite a una matrice nel vostro programma, dovete specificare l'elemento che volete usare, individuandolo con un *indice*. L'indice deve essere tra parentesi, immediatamente dopo il nome della matrice, e può essere una qualsiasi espressione numerica con valore maggiore o uguale a zero. Se necessario, il risultato dell'espressione viene arrotondato al più vicino intero.

Il numero di elementi di una matrice determina il massimo valore di ogni suo indice. Se la matrice non è stata definita in un'istruzione DIM, il massimo valore di ogni indice è 10. Il minimo è zero, a meno che si usi l'istruzione OPTION BASE per definire un minimo uguale a 1. Una matrice definita con DIM A(6) ha quindi 7 elementi, a meno che l'indice di valore 0 sia stato eliminato con l'istruzione OPTION BASE 1.

Nell'esempio sulla destra si suppone che la matrice inizi con l'elemento 1 (OPTION BASE 1 alla linea 120)

- linea 130 - Viene definita la matrice T con 25 elementi
- linea 160 - La variabile numerica I viene usata come indice della matrice T. Il valore che I contiene in questo punto individua un elemento di T. Se I=3, verrà sommato il terzo elemento di T.
- linea 200 - L'indice 14 dice al calcolatore di stampare il quattordicesimo elemento di T.
- linea 220 - Il calcolatore calcola l'espressione numerica N+2. Se N vale 15 in questo istante, viene stampato il quindicesimo elemento di T.

Se vi riferite a una matrice con un indice maggiore del numero di elementi che contiene, o con valore zero (nel caso abbiate usato l'istruzione OPTION BASE 1), compare il messaggio "BAD SUBSCRIPT" e il programma si ferma.

Esempi:

```
>NEW

>100 REM DIM E INDICI
>110 S=100
>120 OPTION BASE 1
>130 DIM T(25)
>140 FOR I=1 TO 25
>150 READ T(I)
>160 A=S+T(I)
>170 PRINT A;
>180 NEXT I
>190 PRINT::
>200 PRINT T(14)
>210 INPUT "SCRIVI NUMERO TRA 1
E 23: ";N
>220 PRINT T(N+2)
>230 DATA 12,13,43,45,65,76,7
 8,98,56,34,23,21,100,333,222
 ,111,444,666,543,234,89,765,
 90,101,345
>240 END
>RUN
  112  113  143  145  165
  176  178  198  156  134
  123  121  200  433  322
  211  544  766  643  334
  189  865  190  201  445


  333

SCRIVI NUMERO TRA 1 E 23: 14
  111

 ** DONE **
```

OPTION BASE

OPTION BASE $\begin{Bmatrix} 0 \\ 1 \end{Bmatrix}$

L'istruzione OPTION BASE vi consente di definire un limite inferiore uguale a 1 (invece che 0), per gli indici delle matrici. Se volete lasciarlo uguale a 0, non avete bisogno di usare questa istruzione.

L'istruzione OPTION BASE deve avere numero di linea più basso di qualsiasi istruzione DIM e di qualsiasi istruzione che faccia riferimento a un elemento di una matrice. In un programma ci può essere *una sola* OPTION BASE, ed è valida per tutte le matrici del programma. Perciò non potete avere una matrice con indice che inizia da zero, e una che inizia da 1 nello stesso programma.

Se usate un intero diverso da 1 o da 0 nell'istruzione OPTION BASE, compare il messaggio "INCORRECT STATEMENT" e il programma si ferma.

Esempi:

```
>NEW

>100 OPTION BASE 1
>110 DIM(5,5,5)
>120 X(1,0,1)=3
>130 PRINT X(1,0,1)
>140 END
>RUN

 * BAD SUBSCRIPT IN 120

>100 ENTER
>RUN
  3

 ** DONE **
```

Sottoprogrammi

Introduzione

I sottoprogrammi sono parti di programma a sè stanti, all'interno di un programma principale. In genere eseguono operazioni particolari, come stampe, calcoli complessi o lettura di dati in una matrice. L'uso di sottoprogrammi vi consente di scrivere un insieme di istruzioni solo una volta, ed eseguirle in qualsiasi punto del programma con l'istruzione GOSUB.

L'istruzione GOSUB funziona in modo simile alla GOTO, in quanto provoca il salto del programma al *numero di linea* specificato. A differenza della GOTO, però, il programma "ricorda" la posizione in cui è avvenuto il salto, e vi ritorna quando ha finito di eseguire il sottoprogramma. L'ultima istruzione di un sottoprogramma deve essere una RETURN. Immediatamente prima del sottoprogramma si mette in genere una STOP o un salto incondizionato, per evitare che il calcolatore lo esegua per sbaglio. I sottoprogrammi devono essere eseguiti solo con l'istruzione GOSUB e possono essere messi in qualsiasi punto del programma principale.

GOSUB

{ GOSUB / GO SUB } *numero di linea*

L'istruzione GOSUB si usa in coppia con la RETURN per passare dal programma principale a un sottoprogramma, eseguirlo e tornare alla linea immediatamente successiva alla GOSUB stessa. Quando il calcolatore esegue una GOSUB, "ricorda" il numero di linea dell'istruzione successiva nel programma principale, in modo da tornare a questo punto quando incontra la RETURN nel sottoprogramma.

(Lo spazio tra GO e SUB è opzionale.)

Esempi:

```
>NEW

>100 REM STAMPA TABELLA VALORI
 TABELLA TRIPLO VALORI
>110 FOR X=1 TO 4
>120 FOR Y=1 TO 7
>130 I(X,Y)=INT(30*RND)+1
>140 NEXT Y
>150 NEXT X
>160 PRINT "PRIMA TABELLA":
>170 GOSUB 260
>180 FOR X=1 TO 4
>190 FOR Y=1 TO 7
>200 I(X,Y)=3*I(X,Y)
>210 NEXT Y
>220 NEXT X
>230 PRINT "3 VOLTE I VALORI I
 LA PRIMA TABELLA":
>240 GOSUB 260
>250 STOP
>260 REM SOTTOPROGRAMMA STAMP
>270 FOR X=1 TO 4
>280 FOR Y=1 TO 7
>290 PRINT I(X,Y);
>300 NEXT Y
>310 PRINT
>320 NEXT X
>330 PRINT
>340 RETURN
>RUN
 PRIMA TABELLA

  16  12  17  12  8  17  8
  18  22  1  29  16  14  11
  5  25  22  4  24  11  24
  26  21  18  2  12  20  15

3 VOLTE I VALORI DELLA PRIMA T
ELLA

  48  36  51  36  24  51  24
  54  66  3  87  48  42  33
  15  75  66  12  72  33  72
  78  63  54  6  36  60  45

 ** DONE **
```

GOSUB

Potete passare da un sottoprogramma a un altro, eseguire quest'ultimo, tornare al primo, eseguirlo a sua volta e tornare infine al programma principale. Dovete solo usare in modo opportuno le istruzioni GOSUB e RETURN, facendo attenzione che il calcolatore possa sempre ricordare dove tornare.

Nell'esempio a destra, il programma salta al sottoprogramma 1 quando arriva alla linea 500. Dal sottoprogramma 1 salta al 2 quando arriva alla linea 730. Quando incontra il RETURN del secondo sottoprogramma, torna alla linea 740, finisce la sua esecuzione, ritorna al programma principale e prosegue fino alla linea 600.

Se l'istruzione GOSUB trasferisce il controllo a un *numero di linea* inesistente, compare il messaggio "BAD LINE NUMBER" e il programma si ferma. Se la GOSUB trasferisce il programma al suo *numero di linea*, compare il messaggio "MEMORY FULL".

Esempi:

```
>NEW

>100 REM SOTTOPROGRAMMI CONCATEN
ATI
>110 REM PROGRAMMA PRINCIPALE
   .
   .
>500 GOSUB 700
>510   .
       .
       .
>600 STOP
>700 REM SOTTOPROGRAMMA 1
       .
       .
>730 GOSUB 800
>740   .
       .
       .
>790 RETURN
>800 REM SOTTOPROGRAMMA 2
       .
       .
>850 RETURN
```

```
>NEW

>100 X=12
>110 Y=23
>120 GOSUB 120
>130 PRINT Z
>140 STOP
>150 REM SOTTOPROGRAMMA
>160 Z=X+Y*120/5
>170 RETURN
>RUN

 * MEMORY FULL IN 120


>120 GOSUB 150
>RUN
  564

 ** DONE **
```

RETURN

RETURN

L'istruzione RETURN si usa con la GOSUB per creare una procedura di salto e ritorno in TI BASIC.
Quando il calcolatore incontra una RETURN ritorna alla linea di programma immediatamente successiva all'istruzione GOSUB che l'ha fatto saltare a quel sottoprogramma. Potete facilmente sviluppare programmi con sottoprogrammi che richiamano altri sottoprogrammi e tornano poi al punto di partenza. Dovete però stare attenti che a ogni GOSUB corrisponda un RETURN.

Se, durante l'esecuzione di un programma, il calcolatore incontra un RETURN senza avere prima eseguito una GOSUB, compare il messaggio "CAN'T DO THAT".

Esempi:

```
>NEW

>100 FOR I=1 TO 3
>110 GOSUB 150
>120 PRINT "I=";I
>130 NEXT I
>140 STOP
>150 REM SOTTOPROGRAMMA
>160 FOR X=1 TO 2
>170 PRINT "X=";X
>180 NEXT X
>190 RETURN
>RUN
X= 1
X= 2
I= 1
X= 1
X= 2
I= 2
X= 1
X= 2
I= 3

** DONE **
```

ON-GOSUB

ON *espressione numerica* { GOSUB, / GO SUB } *numero di linea* [, *numero di linea*]...

L'istruzione ON-GOSUB si usa con il RETURN per dire al calcolatore di eseguire uno tra diversi sottoprogrammi, a seconda del valore dell'*espressione numerica*, e tornare poi al programma principale.

L'*espressione numerica* viene calcolata, e il risultato arrotondato a un intero, se necessario. Questo valore dice al calcolatore quale sottoprogramma (che si trova a un certo *numero di linea*) eseguire. Se è 1, il calcolatore passa a eseguire l'istruzione che si trova al primo *numero di linea* della ON-GOSUB. Se è 2 passa al secondo *numero di linea* e così via.

Il calcolatore "ricorda" il numero della linea successiva alla ON GOSUB, e vi ritorna alla fine del sottoprogramma. Il sottoprogramma deve avere un'istruzione RETURN che dica al calcolatore di tornare a quel punto del programma principale per continuarne l'esecuzione. Se non c'è il RETURN, il programma continua fino alla fine, come se fosse stata eseguita una GOTO invece che una GOSUB.

Esempi:

```
>NEW

>100 INPUT "COD=?":COD
>110 IF COD=9 THEN 290
>120 INPUT "ORE=?":ORE
>130 ON COD GOSUB 170,200,230,26
0
>140 PAGA=QUOTA*ORE+BASE
>150 PRINT "PAGA = ";PAGA
>160 GOTO 100
>170 QUOTA=3000
>180 BASE=2500
>190 RETURN
>200 QUOTA=4000
>210 BASE=5000
>220 RETURN
>230 QUOTA=5000
>240 BASE=7500
>250 RETURN
>260 QUOTA=6000
>270 BASE=10000
>280 RETURN
>290 END
>RUN
 COD=?4
 ORE=?20
 PAGA = 130000
 COD=?2
 ORE=?10
 PAGA = 45000
 COD=?3
 ORE=?15
 PAGA = 82500
 COD=?1
 ORE=?30
 PAGA = 92500

 ** DONE **
```

Se il valore dell'*espressione numerica* è minore di 1 o maggiore del numero di *numeri di linea* della ON-GOSUB, compare il messaggio "BAD VALUE IN xx" e il programma si ferma.

```
>RUN
 COD=?5
 ORE=?10

  * BAD VALUE IN 130
```

Se il *numero di linea non corrisponde a nessuna linea del programma, quando viene eseguita l'istruzione ON-GOSUB compare il messaggio "BAD LINE NUMBER"*

```
>130 ON COD GOSUB 170,200,230,60
0
>RUN
 COD=?4
 ORE=?20

  * BAD LINE NUMBER IN 130
```

Gestione File

Introduzione

Esempi:

Il vostro Home Computer TI ha la capacità di memorizzare su dispositivi esterni programmi e dati sotto forma di file. Potete in seguito richiamare questi file quando volete, e cancellarli quando non vi servono più.

La possibilità di gestire file con il vostro calcolatore è un potente strumento di programmazione. Potete evitare di riscrivere i programmi più usati, conservare informazioni importanti e creare delle procedure per aggiornare i dati che vi servono. Il TI BASIC vi consente diverse procedure gestione-file, con possibilità di accesso sequenziale o diretto, record a lunghezza fissa o variabile e diversi formati per la visualizzazione su video e la rappresentazione interna. Questo paragrafo descrive le istruzioni TI BASIC per la gestione-file, OPEN, CLOSE, INPUT, PRINT e RESTORE. Man mano che usate nuovi dispositivi, studierete sui manuali le caratteristiche dei loro file.

Nota - I nomi dei dispositivi in TI BASIC hanno in genere lettere maiuscole. Per esempio:

CS1
RS232

OPEN

OPEN # *numero file: nome file*[, *organizzazione file*] [, *tipo file*] [, *modo di apertura*] [, *tipo record*] [, *vita file*]

L'istruzione OPEN predispone un programma BASIC a usare file di dati memorizzati su un dispositivo esterno, stabilendo una corrispondenza tra il *numero file* che usate nel programma e il particolare dispositivo su cui si trova il file.

```
>100 OPEN #2:"CS1",SEQUENTIAL
 ,INTERNAL,INPUT,FIXED 128,PE
RMANENT
```

L'istruzione OPEN descrive le caratteristiche di un file, in modo che il calcolatore possa usarlo o crearlo. Per alcuni dispositivi il calcolatore controlla che le caratteristiche del file o del dispositivo siano coerenti con quelle dichiarate nella OPEN. Se non lo sono, o comunque il calcolatore non riesce a trovare o a creare il file, questo non verrà aperto e comparirà un messaggio di errore di Ingresso/Uscita.

Il *numero file* viene dato con il simbolo (#) seguito da un'espressione OPEN. Le altre informazioni possono essere aggiunte in qualsiasi ordine, o anche omesse. Se non le date, il calcolatore assegna d'ufficio alcune caratteristiche standard al file, chiamate "defaults", come vedremo più avanti.

OPEN

■ *numero file* - Tutte le istruzioni TI BASIC che fanno riferimento a un file usano un *numero file* tra 0 e 255 compresi. Questo numero viene assegnato al file tramite la OPEN. Il *numero file* 0 è riservato alla tastiera e al video del vostro calcolatore, e non potete quindi usarlo in un'istruzione di programma. Gli altri numeri possono essere assegnati a vostro piacimento, purchè ogni file ne abbia uno diverso.

Il *numero file* viene dato con il simbolo(#)seguito da un'espressione numerica. Il risultato dell'espressione, arrotondato se necessario all'intero più vicino, deve essere tra 0 e 255 compresi e deve essere diverso da quello di ogni altro file usato nel programma.

Esempi:

```
>100 OPEN #25:"CS1",SEQUENTIA
 L,INTERNAL,INPUT,FIXED,PERMA
 NENT
>110 X=100
>120 OPEN #X+5:"CS2",SEQUENTI
 AL,INTERNAL,OUTPUT,FIXED,PER
 MANENT
```

■ *nome file* - Il *nome file* individua un dispositivo o un file, a seconda del tipo di dispositivo. Ogni dispositivo ha un nome riconosciuto dal calcolatore. Per esempio, i *nome file* dei due registratori a cassette sono "CS1" e "CS2". Col *nome file* nella OPEN dite al calcolatore di usare un particolare file o un particolare dispositivo tutte le volte che nel programma vi si fa riferimento col suo *numero file*. Il *nome file* può essere qualsiasi espressione di stringa. Se è una costante di stringa, deve essere tra virgolette.

```
>130 N=2
>140 OPEN #122:"CS"&STR$(N),S
 EQUENTIAL,INTERNAL,OUTPUT,FI
 XED,PERMANENT
```

Tutte le informazioni circa i *nomi file* associati al Sistema di Memoria a Dischi TI, all'interfaccia RS232 e agli altri dispositivi si trovano nei rispettivi manuali.

■ *organizzazione file* - I file in TI BASIC possono essere ad accesso sequenziale o diretto (random). I record di un file sequenziale vengono letti o scritti in sequenza uno dopo l'altro, dall'inizio alla fine. I record dei file ad accesso diretto, che in TI BASIC si chiamano RELATIVI (RELATIVE), possono essere letti o scritti in qualsiasi ordine, e quindi anche in ordine sequenziale.

```
>100 OPEN #4:"CS2",OUTPUT,INT
 ERNAL,SEQUENTIAL,FIXED
```

Per indicare la struttura logica di un file dovete mettere la parole SEQUENTIAL o RELATIVE nella OPEN. Potete anche specificare, dopo di queste, il numero iniziale di record con un' espressione numerica.

```
>120 OPEN #12:NOME$,RELATIVE
 50,INPUT,FIXED,INTERNAL
```

In mancanza di specifica, il file verrà considerato SEQUENTIAL.

```
>100 OPEN #10:"CS1",OUTPUT,FI
 XED
(IL CALCOLATORE SOTTINTENDE: SEQ
UENTIAL, DISPLAY, PERMANENT)
```

■ *tipo file* - Questo dato definisce il formato dei dati del file - DISPLAY o INTERNAL.

Il tipo DISPLAY si riferisce a caratteri ASCII stampabili, ed è usato in genere quando l'uscita deve essere letta da un uomo e non dal calcolatore. Ogni record di tipo DISPLAY corrisponde di solito a una riga di stampa.

I dati di tipo INTERNAL sono registrati con la rappresentazione interna della macchina, e non sono quindi stampabili. Possono essere letti facilmente dal calcolatore, ma non da un uomo.(Vedi l'istruzione "INPUT" per una spiegazione completa della rappresentazione interna.)

OPEN

Vedrete che il tipo INTERNAL è adatto alla registrazione dei dati su cassette, poichè richiede meno spazio ed è facile assegnargli un formato con l'istruzione PRINT (nel paragrafo "PRINT" è spiegato come si definisce un formato tramite questa istruzione per i record sia di tipo INTERNAL che DISPLAY). Dato che il calcolatore usa al suo interno dati di tipo INTERNAL, un programma i cui file siano di questo tipo è più veloce che nell'altro caso, in quanto i dati non devono essere tradotti da DISPLAY in INTERNAL e viceversa.

Se manca questa specifica, il calcolatore adotta il formato DISPLAY.

- *modo di apertura* - Con questa informazione dite al calcolatore se elaborare i file in modo INPUT, OUTPUT, UPDATE, o APPEND. Se manca viene usato il modo UPDATE.
  - In modo INPUT i file possono essere solo letti.
  - in modo OUTPUT i file possono solo essere scritti. Un file appena creato avrà le caratteristiche descritte nella OPEN e le eventuali "defaults".
  - In modo UPDATE i file possono essere sia letti che scritti. Un'elaborazione normale su un file comporta la lettura di un record, qualche modifica su di esso e la riscrittura sul file del record modificato.
  - In modo APPEND si possono aggiungere record alla fine di un file esistente, ma non si può accedere ai record che già fanno parte del file.

- *tipo record* - Questo dato specifica se i record di un file hanno tutti la stessa lunghezza (FIXED), o sono di lunghezza varabile (VARIABLE). Le parole FIXED e VARIABLE possono essere seguite da un'espressione numerica che specifica la lunghezza massima di un record. Ogni dispositivo ha un suo massimo di lunghezza record, che è riportato nel suo manuale, e che viene assegnata ai file di quel dispositivo, se non ne viene specificata una diversa.

  Un file RELATIVE deve avere record di lunghezza fissa (FIXED), che sarà quella da voi specificata, o, in sua mancanza, quella del dispositivo su cui è registrato.

  I file SEQUENTIAL possono avere record a lunghezza fissa o variabile. Il *tipo record* di defaults è a lunghezza variabile.

  Se i record sono FIXED, ogni record viene completato a destra in modo da raggiungere la lunghezza specificata. Se il dato è in formato DISPLAY, il record viene completato con degli spazi, se è in formato INTERNAL, con degli zeri binari.

- *vita del file* - I file che create con il vostro calcolatore sono considerati PERMANENT (permanenti), non temporanei, e quindi potete fare a meno di dare questa specifica.

Esempi:

```
>100 OPEN #53:NOMES,FIXED,INT
 ERNAL,RELATIVE

(IL CALCOLATORE SOTTINTENDE: UP
ATE)
```

```
>100 OPEN #11:NOMES,INPUT,INT
 ERNAL,SEQUENTIAL,VARIABLE 10
 0

>100 OPEN #75:"CS1",OUTPUT,FI
 XED

(IL CALCOLATORE SOTTINTENDE: SE
UENTIAL,DISPLAY,FIXED LUNGHEZZA
DI 64 CARATTERI)
```

OPEN

Caratteristiche Registratore a Cassette

- *numero file** - qualsiasi numero tra 1 e 255 compresi
- *nome file** - "CS1" o "CS2"
- *organizzazione file* - SEQUENTIAL
- *tipo file* - INTERNAL(consigliato) o DISPLAY
- *modo di apertura** - INPUT o OUTPUT
- *tipo record** - FIXED

*Questa specifica è necessaria

Per i record di file su cassetta, potete specificare una lunghezza qualsiasi fino a 192. Il registratore, d'altra parte, usa record con lunghezza 64, 128 o 192, e completa quindi i vostri record fino a ottenere una di queste lunghezze. Così, se date una lunghezza di 83, di fatto il record avrà 128 posizioni. Se non date nessuna lunghezza avrà 64 posizioni.

Con i registratori a cassetta, il calcolatore non confronta le specifiche della OPEN con le caratteristiche dei file.

Quando viene eseguita un'istruzione di OPEN per un regitratore, sul video compaiono le istruzioni per il suo funzionamento, come vedete nell'esempio sulla destra.

Nota - Per un file di INPUT può essere usato solo "CS1", mentre sia "CS1" che "CS2" possono essere usati per file di OUTPUT.

Esempi:

```
>NEW

>100 OPEN #2:"CS1",INTERNAL,I
 NPUT,FIXED
 .
 .LINEE PROGRAMMA
 .
>290 CLOSE #2
>300 END
>RUN

 * REWIND CASSETTE TAPE  CS1
   THEN PRESS ENTER

 * PRESS CASSETTE PLAY  CS1
   THEN PRESS ENTER
   .
   . IL PROGRAMMA CONTINUA
   .

 * PRESS CASSETTE STOP  CS1
   THEN PRESS ENTER

 ** DONE **
```

CLOSE

CLOSE # *numero file* [:DELETE]

L'istruzione CLOSE "chiude" o interrompe la comunicazione tra un file e un programma. Una volta eseguita, il file "chiuso" non è più accessibile al programma fin quando non lo "aprite" nuovamente con una OPEN. Il calcolatore non associa più a quel file il *numero file* che avete specificato nel programma, e quindi potete assegnare questo numero a un altro file.

Se nell'istruzione CLOSE usate l'opzione DELETE, viene eseguita un'operazione diversa a seconda del dispositivo che state usando. Gli effetti della DELETE sui diversi dispositivi sono descritti nei relativi Manuali.

Se cercate di chiudere un file che non avete precedentemente aperto nel programma, compare il messaggio "FILE ERROR" e il programma si ferma.

Per proteggere i vostri file il calcolatore li chiude automaticamente quando capita un errore che interrompe l'esecuzione del programma. Quando il programma viene interrotto o con un comando BREAK o premendo il tasto **CLEAR**, i file aperti sono automaticamente chiusi *solo* se

- state modificando un programma
- uscite dal BASIC con un comando BYE
- rilanciate il programma con un comando RUN
- date il comando NEW

Nota - Se usate il comando QUIT per uscire dal programma, i file aperti *NON* vengono chiusi e i loro dati vengono persi. Se dovete uscire da un programma di elaborazione-file prima di terminare la sua esecuzione, seguite le istruzioni seguenti per non perdere i dati.

- Premete **CLEAR** finchè il calcolatore vi invia il messaggio "BREAKPOINT AT xx". Può darsi che ci metta alcuni secondi.
- Premete BYE quando il cursore ricompare sul video.

Esempi:

```
>NEW

>100 OPEN #6:"CS1",SEQUENTIAL
 ,INTERNAL,INPUT,FIXED
>110 OPEN #25:"CS2",SEQUENTIA
 L,INTERNAL,OUTPUT,FIXED
 .
 .LINEE PROGRAMMA

>200 CLOSE #6:DELETE
>210 CLOSE #25
>220 END
```

CLOSE

Caratteristiche del registratore a cassette

Quando il calcolatore esegue un'istruzione CLOSE per un registratore, sul video compaiono le istruzioni per il suo funzionamento, come si vede nell'esempio sulla destra.

Anche se usate l'opzione DELETE, l'unica operazione che ha luogo è la chiusura dei file.

Esempi:

```
>NEW

>100 OPEN #24:"CS1",INTERNAL,
 INPUT,FIXED
>110 OPEN #19:"CS2",INTERNAL,
OUTPUT,FIXED
 .
 .LINEE PROGRAMMA
 .
>200 CLOSE #24
>210 CLOSE #19
>220 END
>RUN

 * REWIND CASSETTE TAPE  CS1
   THEN PRESS ENTER

 * PRESS CASSETTE PLAY   CS1
   THEN PRESS ENTER

 * REWIND CASSETTE TAPE  CS2
   THEN PRESS ENTER

 * PRESS CASSETTE RECORD CS2
   THEN PRESS ENTER
   .
   .IL PROGRAMMA CONTINUA
   .

 * PRESS CASSETTE STOP   CS1
   THEN PRESS ENTER

 * PRESS CASSETTE STOP   CS2
   THEN PRESS ENTER

** DONE **
```

INPUT

INPUT # *numero file* [, REC espressione numerica] [: *lista variabili*]

(Vedere il paragrafo "Istruzioni di Ingresso/Uscita")

Questo tipo di istruzione di INPUT vi consente di leggere dati da un dispositivo esterno, e può essere usata solo con file aperti in modo INPUT o UPDATE. Il *numero file* che compare nell'istruzione deve essere quello di un file già aperto. Può essere usato il *numero file* 0, che si riferisce alla tastiera. In questo caso l'istruzione INPUT viene eseguita come descritto nel paragrafo"Istruzioni di Ingresso/Uscita", ma non potete specificare un messaggio di prompt.

La *lista variabili* contiene le variabili a cui vengono assegnati i valori quando è eseguita l'istruzione di INPUT. I nomi delle variabili sono separati da virgole e possono essere numerici o di stringa.

Esempi:

```
>NEW

>100 OPEN #13:"CS1",SEQUENTIA
 L,DISPLAY,INPUT,FIXED
>110 INPUT #13:A,B,C$,D$,X,Y,
 Z$
>120 IF A=99 THEN 150
>130 PRINT A;B:C$:D$:X;Y:Z$
>140 GOTO 110
>150 CLOSE #13
>160 END
>RUN

--I DATI MEMORIZZATI SUL NASTRO
VENGONO STAMPATI SULLO SCHERMO

 ** DONE **
```

Assegnazione dei valori alla lista delle variabili

Quando il calcolatore legge dei record da un file, li sistema provvisoriamente in un'area di memoria chiamata "buffer di I/O"(Input/Output). Per ogni file aperto con un certo *numero file* è previsto un beffer di I/O. I dati di questo buffer vengono assegnati alle variabili della *lista variabili* da sinistra verso destra. Quando una *lista variabili* è riempita con i valori corrispondenti, i dati che restano nel buffer vengono ignorati, a meno che l'istruzione INPUT finisca con una virgola. In questo caso, l'operazione di ingresso resta in sospeso (vedi il paragrafo "Ingresso in sospeso"). sospeso").

```
>NEW

>100 OPEN #13:"CS1",SEQUENTIA
 L,DISPLAY,INPUT,FIXED 64
>110 INPUT #13:A,B,C,D
 .
 .LINEE PROGRAMMA
 .
>290 CLOSE #13
>300 END
>RUN

--PRIMO RECORD DI INPUT: 22,77.5
6,92

--RISULTATI:
    A=22  B=77  C=56  D=92
```

Se la *lista variabili* nell'istruzione INPUT è più lunga del numero di dati del record letto, il calcolatore passa al record successivo, e completa la *lista variabili* con dati di quest'ultimo, come si vede nell'esempio a destra.

Quando esegue un'istruzione di INPUT il calcolatore si comporta diversamente a seconda che i dati siano in formato INTERNAL o DISPLAY.

```
>NEW

>100 OPEN #13:"CS1",SEQUENTIA
 L,DISPLAY,INPUT,FIXED 64
>110 INPUT #13:A,B,C,D,E,F,G
 .
 .LINEE PROGRAMMA
 .
>400 END

--PRIMO RECORD DI INPUT: 22,33.5
--SECONDO RECORD DI INPUT: 405,9
2
--TERZO RECORD DI INPUT: -22,110
23
--QUARTO RECORD DI INPUT: 99,100

--RISULTATI:

  A=22  B=33.5  C=405  D=92

  E=-22  F=11023  G=99
```

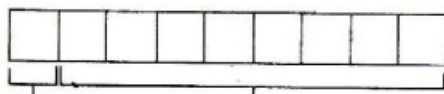
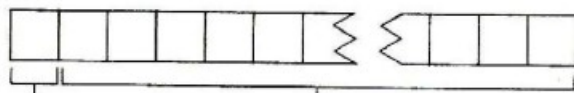
INPUT

Dati di tipo DISPLAY

I dati di tipo DISPLAY hanno lo stesso formato di quelli inseriti da tastiera. Il calcolatore riconosce la lunghezza di ogni dato del record tramite la virgola che lo separa dal successivo.

Ogni dato di un record di tipo DISPLAY viene controllato per assicurarsi che a variabili numeriche siano assegnati valori numerici, come si vede nell'esempio sulla destra per il record 1. Se il tipo del dato non è conforme a quello della variabile, come nel caso del record 2 (JG non è un valore numerico), si ha un INPUT ERROR e il programma si ferma.

Dati di tipo INTERNAL

I dati di tipo INTERNAL hanno il seguente formato

Il calcolatore riconosce la lunghezza di un dato di tipo INTERNAL dall'indicatore di lunghezza che occupa la prima posizione del dato.

Sui dati di tipo INTERNAL vengono eseguiti alcuni controlli. Quelli numerici devono avere 9 posizioni (8 cifre più una per la lunghezza) e devono essere una rappresentazione corretta di un numero in virgola mobile, altrimenti si ha un INPUT ERROR e il programma si ferma.

Per i record di tipo INTERNAL a lunghezza fissa, una lettura effettuata al di là dei dati registrati in ogni record dà degli 0 binari. Se cercate di assegnare questi caratteri a una variabile numerica si ha INPUT ERROR. Se invece li assegnate a una variabile di stringa, questa risulta di lunghezza nulla.

Esempi:

```
>NEW

>100 OPEN #13:"CS1",SEQUENTIA
 L,DISPLAY,INPUT,FIXED 64
>110 INPUT #13:A,C,STATO$,D$,
 X,Y

--RECORD DI INPUT: 22, 97.6, TEX
AS,"LICENZA AUTO",22000,-.07

--ALTRO RECORD DI INPUT: JG, 22
 TEXAS, TASSA DI PROPRIETA', 42,
 15
```

L'istruzione INPUT con i file RELATIVE

(Vedi l'istruzione "OPEN" per la descrizione dei file RELATIVE.)

I file RELATIVE possono essere letti sia in modo sequenziale che diretto. Il calcolatore aggiorna un contatore interno per puntare il record che deve essere elaborato. Il primo record di un file è il record 0. Il contatore quindi parte da zero, e viene incrementato di 1 dopo ogni accesso al file, sia per leggere che per scrivere un record. Nell'esempio a destra, il file viene letto sequenzialmente.

Il contatore interno può essere modificato con l'opzione REC. L'espressione numerica che segue la parola REC individua con il suo valore un numero di record nel file. Quando il calcolatore esegue un'istruzione di INPUT con l'opzione REC, legge dal file il record specificato e lo pone nel buffer di I/O. L'opzione REC si può usare solo con i file RELATIVE. Nell'esempio a destra si usa un file RELATIVE ad accesso diretto, con l'opzione REC.

Dovete usare l'opzione REC se leggete e scrivete record da uno stesso file. Infatti il contatore interno viene incrementato ogni volta che leggete o scrivete un record dallo stesso file, e se non usate la REC vi può capitare di saltare dei record e scrivere su record diversi da quelli che vorreste, come mostrato nell'esempio sulla destra.

Se il contatore interno punta a un record oltre la fine del file, quando il calcolatore tenta di accedersi si ha un INUT ERROR.

Esempi:

```
>NEW
>100 OPEN #4:NOME$,RELATIVE,I
 NTERNAL,INPUT,FIXED 64
>110 INPUT #4:A,B,C$,D$,X
 .
 .LINEE PROGRAMMA
 .
>200 CLOSE #4
>210 END
```

```
>NEW

>100 OPEN #6:NOME$,RELATIVE,I
 NTERNAL,UPDATE,FIXED 72
>110 INPUT K
>120 INPUT #6,REC K:A,B,C$,D$
 .
 .LINEE PROGRAMMA
 .
>170 PRINT #6,REC K:A,B,C$,D$
 .

 .LINEE PROGRAMMA
 .
>300 CLOSE #6
>310 END
```

```
>NEW

>100 OPEN #3:NOME$,RELATIVE,I
 NTERNAL,UPDATE,FIXED
>110 FOR I=1 TO 10
>120 INPUT #3:A$,B$,C$,X,Y
 .
 .LINEE PROGRAMMA
 .
>230 PRINT #3:A$,B$,C$,X,Y
>240 NEXT I
>250 CLOSE #3
>20 END
>RUN

--LINEA 120 LEGGE IL RECORD: 0,2
,4,6,8....

--LINEA 230 SCRIVE IL RECORD: 1,
3,5,7,9,...
```

Ingresso in sospeso

Quando viene eseguita un'istruzione di INPUT che termina con una virgola, si crea una situazione di "ingresso in sospeso". La successiva INPUT dallo stesso file causa una delle seguenti azioni

- Se la INPUT successiva non ha l'opzione REC, il calcolatore preleva i dati dal buffer di I/O a partire da dove ha finito la INPUT precedente.
- Se la INPUT successiva usa l'opzione REC, il calcolatore annulla la condizione di "ingresso in sospeso" e legge nel buffer di I/O il record specificato.

Se, in condizione di "ingresso in sospeso" viene eseguita una PRINT sullo stesso file, la condizione viene ignorata e la PRINT viene eseguita normalmente.

Se create una condizione di "ingresso in sospeso" con un *numero file* 0, compare il messaggio "INCORRECT STATEMENT" e il programma si ferma.

Esempi:

```
>NEW

>100 INPUT #0:A,B,
>110 PRINT A;B
>120 GOTO 100
>RUN
 ?
 * INCORRECT STATEMENT
   IN 100
```

Fine del File(End of File)

In una elaborazione sequenziale, dovete segnalare al calcolatore che è stata raggiunta la fine del file, per evitare errori. Per facilitarvi questo compito il TI BASIC vi fornisce la funzione **EOF** (End of File). Mettete l'istruzione EOF subito prima della INPUT che legge un file sequenziale, per fare in modo che il calcolatore finisca di leggere il file quando non ci sono più dati. In genere la EOF si usa per saltare a una routine di chiusura del file.

```
>NEW

>100 OPEN #5:NOME$,SEQUENTIAL
 ,INTERNAL,INPUT,FIXED
>110 IF EOF(5) THEN 150
>120 INPUT #5:A,B
>130 PRINT A;B
>140 GOTO 110
>150 CLOSE #5
>160 END
```

La funzione EOF non si può usare con i file RELATIVE e con alcuni dispositivi esterni. In questi casi dovete creare voi la procedura per determinare la fine del file.

Un metodo molto comune consiste nell'usare l'ultimo record del file come indicatore della fine del file. Questo record viene chiamato "dummy" (fittizio) perchè i dati che contiene non sono significativi, se non per indicare la fine del file. Potete, ad esempio, riempirlo di "9" e, ogni volta che leggete un record, controllare il suo contenuto. Se sono tutti "9", sapete che siete arrivati alla fine del file e potete passare alla routine di chiusura.

Il primo esempio sulla destra crea un record "dummy". Nel secondo, questo record viene usato come indicatore della fine del file.

Caratteristiche del registratore a cassette

- Sui registratori a cassetta non possono essere usati file RELATIVE.
- La funzione EOF (End-of-File) non può essere usata sui file di un registratore a cassette
- La lunghezza dei record può essere al massimo 192 (vedi l'istruzione "OPEN")
- Solo l'unità 1 (CS1) può essere usata per leggere dati.

Esempi:

```
>NEW

>100 OPEN #2:"CS1",SEQUENTIAL
 ,FIXED,OUTPUT,INTERNAL
>110 READ A,B,C
>120 IF A=99 THEN 180
>130 E=A+B+C
>140 PRINT A;B;C;E
>150 PRINT #2:A,B,C,E
>160 GOTO 110
>170 DATA 5,10,15,10,20,30,10
 0,200,300,99,99,99
>180 PRINT #2:99,99,99,99
>190 CLOSE #2
>200 END
>RUN

 * REWIND CASSETTE TAPE    CS1
   THEN PRESS ENTER

 * PRESS CASSETTE RECORD   CS1
   THEN PRESS ENTER
 5  10  15  30
 10  20  30  60
 100  200  300  600

 * PRESS CASSETTE STOP     CS1
   THEN PRESS ENTER

 ** DONE **
```

```
>NEW

>100 OPEN #1:"CS1",INTERNAL,I
 NPUT,FIXED
>110 INPUT #1:A,B,C,E
>120 IF A=99 THEN 160
>130 F=A*E
>140 PRINT A;B;C;E;F
>150 GOTO 110
>160 CLOSE #1
>170 END
>RUN

 * REWIND CASSETTE TAPE    CS1
   THEN PRESS ENTER

 * PRESS CASSETTE PLAY     CS1
   THEN PRESS ENTER
 5  10  15  30  150
 10  20  30  60  600
 100  200  300  600  6000

 * PRESS CASSETTE STOP     CS1
   THEN PRESS ENTER

 ** DONE **
```

EOF (End-of-File)

EOF*(espressione numerica)*

Questa funzione determina se è stata raggiunta la fine di un file su di un dispositivo esterno. Il valore dell'argomento, ottenuto calcolando l'*espressione numerica* con le regole note, specifica il numero di un file aperto (vedi "OPEN").

La funzione assume valori diversi a seconda della posizione del file.

Valore	*Posizione*
0	Il file non è alla fine
+1	Fine logica del file
—1	Fine fisica del file

La fine logica di un file si raggiunge quando tutti i record sono stati elaborati. La fine fisica del file si raggiunge quando non c'è più spazio per il file.

Questa funzione e l'esempio sulla destra non si possono usare sui registratori a cassetta. Il suo utilizzo su altri dispositivi esterni è illustrato nei relativi manuali.

Esempi:

```
>NEW

>100 OPEN #2:NOME$,SEQUENTIAL
 ,INTERNAL,INPUT, FIXED
>110 IF EOF(2) THEN 160
>120 REM SE EOF VALE ZERO
>130 INPUT #2:A,B,C
>140 PRINT A;B;C
>150 GOTO 110
>160 CLOSE #2
>170 END
```

PRINT

PRINT # *numero file* [, *REC espressione numerica*] [: *lista di stampa*]

(Vedi il paragrafo "Istruzioni di Ingresso/Uscita" per una descrizione dell'uso della PRINT per stampare sul video.)

Questa forma dell'istruzione PRINT consente di scrivere dati su un dispositivo esterno, e può essere usata solo con file aperti in modo OUTPUT, UPDATE, o APPEND.Il *numero file* deve essere quello di un file aperto.

Quando il calcolatore esegue una PRINT, trasferisce i dati in un'area di memoria temporanea, detta buffer di I/O. C'è un buffer di I/O per ogni file aperto. Se la PRINT non termina con un separatore(virgola, punto e virgola o due punti), il record viene scritto immediatamente dal buffer di I/O nel file. Se invece la PRINT finisce con un separatore, si ha una condizione di "stampa in sospeso"(vedi il paragrafo "Stampa in sospeso".)

Vi diamo qui di seguito le istruzioni per creare la *lista di stampa*. Per ottenere delle linee di stampa, la *lista di stampa* è la stessa di quella descritta nel paragrafo "Istruzioni di Ingresso/Uscita". Potete usare sia il formato INTERNAL che quello DISPLAY. Però, poichè questi file sono letti solo dal calcolatore è più facile e più efficiente quello INTERNAL.

L'istruzione PRINT con dati di tipo INTERNAL

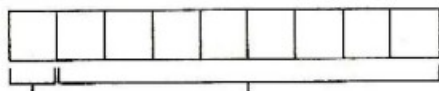
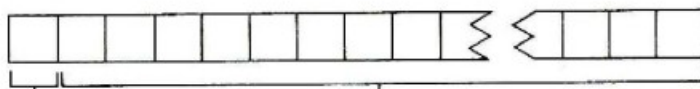
La *lista di stampa* è costituita da espressioni numeriche o di stringa, separate da virgole, punti e virgola e due punti. Per dati di tipo INTERNAL i diversi separatori hanno tutti lo stesso effetto. Semplicemente separano un dato dall'altro, senza indicare eventuali spaziature.

Esempi:

```
>NEW

>100 OPEN #5:"CS1",SEQUENTIAL
 ,INTERNAL,OUTPUT,FIXED
 .
 .LINEE PROGRAMMA
 .
>170 PRINT #5:A,B,C$,D$
 .
 .LINEE PROGRAMMA
 .
>200 CLOSE #5
>210 END
```

```
>NEW

>100 OPEN #6:"CS2",SEQUENTIAL
 ,DISPLAY,OUTPUT,FIXED
 .
 .LINEE PROGRAMMA
 .
>170 PRINT #6:A;",";B;",";C$;
 ",";D$
 .
 .LINEE PROGRAMMA
 .
>200 CLOSE #6
>210 END
```

PRINT

Quando i dati di una *lista di stampa* sono stampati su un dispositivo esterno in formato INTERNAL, hanno le seguenti caratteristiche.

Nell'esempio sulla destra, la lunghezza complessiva dei dati registrati in formato INTERNAL è di 65 posizioni. Ogni variabile numerica occupa 9 posizioni. A$ ha 18 caratteri (linea 110) più una posizione che contiene la lunghezza della stringa. B$ ha 9 caratteri (linea 120) più 1. Se i valori di A$ e B$ cambiano durante il programma, la loro lunghezza varia di conseguenza in base al valore che contengono quando il record viene scritto sul file. Perciò, quando pensate il vostro record, dovete prevedere tutti i possibili valori che ogni variabile può contenere e definire la sua lunghezza come la massima possibile.

Quando usate record a lunghezza fissa, il calcolatore completa, se necessario, quelli di tipo INTERNAL con degli zeri binari, per raggiungere la lunghezza assegnata.

I record non possono essere più lunghi della lunghezza, specificata o di default, del dispositivo che state usando. Se questo accade con record di tipo INTERNAL, compare il messaggio "FILE ERROR IN xx".

```
>NEW

>100 OPEN #5:"CS1",SEQUENTIAL
 ,INTERNAL,OUTPUT,FIXED 128
>110 A$="TEXAS INTRUMENTS "
>120 B$="COMPUTER "
>130 READ X,Y,Z
>140 IF X=99 THEN 190
>150 A=X*Y*Z
>160 PRINT #5:A$,X,Y,Z,B$,A
>170 GOTO 130
>180 DATA 5,6,7,1,2,3,10,20,3
 0,20,40,60,1.5,2.3,7.6,99,99
 ,99
>190 CLOSE #5
>200 END
>RUN

 * REWIND CASSETTE TAPE  CS1
   THEN PRESS ENTER

 * PRESS CASSETTE RECORD CS1
   THEN PRESS ENTER

--I DATI VENGONO SCRITTI SUL NAS
TRO

 * PRESS CASSETTE STOP   CS1
   THEN PRESS ENTER

 ** DONE **
```

PRINT

L'istruzione PRINT con dati di tipo DISPLAY

Anche se è meglio usare il formato INTERNAL quando i dati devono essere letti dal calcolatore, può capitare talvolta di dover usare il formato DISPLAY. Vi diamo alcuni consigli importanti da seguire in questo caso.

- I record vengono creati in base alle specifiche date nell'istruzione PRINT(vedi il paragrafo "Istruzioni di Ingresso/Uscita".)
- Se la *lista di stampa* costituisce un record di lunghezza maggiore di quella specificata o di default del dispositivo usato, l'ultimo elemento aggiunto diventa il primo del record successivo. Se poi un singolo elemento da solo supera la lunghezza del record, viene suddiviso in tanti record quanti sono necessari per contenerlo tutto. L'esecuzione del programma continua regolarmente e non viene inviato alcun messaggio.
- Per poter rileggere in seguito file di tipo DISPLAY creati con la PRINT, i dati devono avere lo stesso aspetto di quando li inserite da tastiera. Perciò devono esserci le virgole di separazione e le virgolette necessarie in genere nella INPUT. Questi segni non vengono inseriti automaticamente quando viene eseguita l'istruzione di PRINT, ma devono comparire come elementi della *lista di stampa*, come nella linea 170 dell'esempio sulla destra.

```
>NEW

>100 OPEN #10:"CS1",SEQUENTIA
 L,DISPLAY,OUTPUT,FIXED 128
 .
 .LINEE PROGRAMMA
 .
>170 PRINT #10:"""";A$;"""",";
 X;",";Y;",";Z;",""";B$;"""","
 ;A
 .
 .LINEE PROGRHMMA
 .
>300 CLOSE #10
>310 END
```

- I dati numerici NON hanno lunghezza fissa, a differenza di quelli con formato INTERNAL. Nei file di tipo DISPLAY, la lunghezza di un dato numerico è la stessa che avrebbe se fosse visualizzato sul video con un'istruzione PRINT o DISPLAY (compresi il segno, il punto decimale, l'esponente, gli spazi, etc.) Per esempio, il numero di posizioni necessarie per stampare 1.35E-10 è dieci.

Stampa in sospeso

Se l'istruzione PRINT non finisce con un separatore, provoca la scrittura di un record sul file. In caso contrario si genera una condizione di stampa "in sospeso", e quando si incontra la successiva PRINT per lo stesso file, accade una delle seguenti cose:

- Se la successiva PRINT non ha l'opzione REC, il calcolatore inserisce i dati nel buffer di I/O subito dopo quelli che ci sono già.
- Se la successiva PRINT ha l'opzione REC, il calcolatore scrive il record "in sospeso" nella posizione del file indicata dal contatore interno ed esegue la PRINT successiva normalmente.

Se in condizione di stampa "in sospeso" si incontra una INPUT per lo stesso file, il record "in sospeso" viene scritto nel file alla posizione indicata dal contatore interno, e questo viene incrementato. La INPUT viene poi eseguita regolarmente. Se in condizione di stampa "in sospeso" il file viene chiuso o riportato all'inizio, il record "in sospeso" viene scritto prima che queste operazioni siano effettuate.

Caratteristiche del registratore a cassette

- Potete specificare una lunghezza record fino a 192 posizioni.
- Potete elaborare solo file SEQUENTIAL, non quelli RELATIVE.

RESTORE

RESTORE # *numero file* [, *REC espressione numerica*]

(Per la descrizione dell'utilizzo della RESTORE con la READ e la DATA, vedi il paragrafo "Istruzioni di Ingresso/Uscita".)

L'istruzione RESTORE riposiziona un file già aperto sul record iniziale (vedi il primo esempio sulla destra), o su un record specificato, se il file è RELATIVE (vedi il secondo esempio sulla destra.)

Se il *numero file* specificato nella RESTORE corrisponde a un file non ancora aperto, compare il messaggio "FILE ERROR IN xx".

L'opzione REC si può usare solo con file RELATIVE. L'*espressione numerica* che segue la REC viene calcolata, e il suo valore fa da puntatore a un particolare record del file. Se usate la RESTORE con un file RELATIVE, senza l'opzione REC, il file viene posizionato sul record 0.

Se c'è una condizione di stampa "in sospeso", il record viene scritto sul file prima di eseguire la RESTORE. Se c'è una condizione di ingresso "in sospeso", i dati nel buffer di I/O sono persi.

I file RELATIVE non possono essere usati su registratori a cassetta.

Esempi:

```
>NEW

>100 OPEN #2:"CS1",SEQUENTIAL
 ,INTERNAL,INPUT,FIXED 64
>110 INPUT #2:A,B,C$,D$,X
 .
 .LINEE PROGRAMMA
 .
>400 RESTORE #2
>410 INPUT #2:A,B,C$,D$,X
 .
 .LINEE PROGRAMMA
 .
>500 CLOSE #2
>510 END
```

```
>NEW

>100 OPEN #4:NOME$,RELATIVE,I
 NTERNAL,UPDATE,FIXED 128
>110 INPUT #4:A,B,C
 .
 .LINEE PROGRAMMA
 .
>200 PRINT #4:A,B,C
 .
 .LINEE PROGRAMMA
 .
>300 RESTORE #4,REC 10
>310 INPUT #4:A,B,C
 .
 .LINEE PROGRAMMA
 .
>400 CLOSE #4
>410 END
```

Messaggi di Errore

I. Ingresso di una Linea

*** BAD LINE NUMBER**

1. Il numero della linea inserita o quello a cui fa riferimento è uguale a 0 o maggiore di 32767
2. La RESEQUENCE genera un numero di linea maggiore di 32767

*** BAD NAME**

1. Il nome di una variabile ha più di 15 caratteri

*** CAN'T CONTINUE**

1. E' stato dato un comando CONTINUE senza che il programma fosse in un punto di arresto, oppure il programma è stato modificato durante un punto di arresto

*** CAN'T DO THAT**

1. Avete usato come comando una delle seguenti istruzioni di programma - DATA-,DEF, FOR, GOTO, GOSUB, IF, INPUT, NEXT, ON ,OPTION, RETURN
2. Avete usato come istruzioni di programma (con numero di linea) uno dei seguenti comandi - BYE, CONTINUE, EDIT, LIST, NEW, NUMBER, OLD, RUN, SAVE
3. Avete dato uno dei comandi LIST, RUN, SAVE senza programma in memoria

*** INCORRECT STATEMENT**

1. Su una riga ci sono due nomi di variabili senza un corretto separatore (ABC A o A$A)
2. Una variabile è seguita da una costante numerica senza un corretto separatore (N 257)
3. In una stringa mancano le virgolette finali
4. Nei comandi LIST, NUMBER o RESEQUENCE ci sono separatori non validi tra i numeri
5. Caratteri non validi seguono i comandi CONTINUE, LIST, NUMBER, RESEQUENCE o RUN
6. Un comando non è la prima parola di una linea
7. Il nome di un dispositivo non è seguito dai due punti in un comando LIST

*** LINE TOO LONG**

1. La linea di ingresso è troppo lunga per il buffer di ingresso

*** MEMORY FULL**

1. La linea inserita è troppo lunga per la memoria disponibile
2. Aggiungendo una linea a un programma si è superata la capacità di memoria disponibile

II. Generazione della Tavola dei Simboli

Quando date il comando RUN, e prima che venga eseguita qualsiasi linea, il calcolatore analizza il programma per creare una *tavola dei simboli*, cioè un'area di memoria in cui vengono memorizzate variabili, matrici, funzioni, etc. Durante questa operazione il calcolatore riconosce alcuni errori del programma, e stampa un messaggio con il numero della linea in errore (per esempio - *BAD VALUE IN 100). Gli errori che vedremo in questo paragrafo differiscono da quelli del paragrafo III in quanto il colore del video resta ciano finchè non è stata creata la tavola dei simboli. Poichè fino a questo punto non è stata eseguita alcuna istruzione, i valori della *tavola dei simboli* sono zero(per i numeri) e di lunghezza zero (per le stringhe).

*** BAD VALUE**

1. Una dimensione di una matrice è maggiore di 32767
2. Una dimensione di una matrice è zero, con OPTION BASE=1

*** CAN'T DO THAT**

1. Nel programma c'è più di una OPTION BASE
2. L'istruzione OPTION BASE ha un numero di linea più alto di un'istruzione che definisce una matrice

*** FOR-NEXT ERROR**

1. Nel programma non c'è lo stesso numero di FOR e di NEXT

Messaggi di Errore

*** INCORRECT STATEMENT**

DEF

1. In un'istruzione DEF manca la ")" dopo un parametro
2. In un'istruzione DEF manca il segno "="
3. Un parametro in un'istruzione DEF non ha un nome di variabile corretto

DIM

4. L'istruzione DIM non ha dimensioni o ne ha più di tre
5. Una dimensione di un'istruzione DIM non è un numero
6. Una dimensione di un'istruzione DIM non è seguita da una virgola o da una ")"
7. Il *nome matrice* di un'istruzione DIM non è un nome di variabile corretto
8. L'indice di una matrice non è seguito da ")"

OPTION BASE

9. OPTION non è seguita da BASE
10. OPTION BASE non è seguita da 0 o 1

*** MEMORY FULL**

1. La dimensione di una matrice è troppo grande
2. Non c'è sufficiente memoria per memorizzare una variabile o una funzione

*** NAME CONFLICT**

1. Avete assegnato lo stesso nome a più di una matrice (DIM A(5), A(2,7))
2. Avete assegnato lo stesso nome a una matrice e a una variabile semplice
3. Avete assegnato lo stesso nome a una variabile e a una funzione
4. Un riferimento a una matrice ha un numero di dimensioni diverso da quello della matrice (B = A (2,7) + 2, PRINT A (5)).

III. Esecuzione di un Programma

Durante l'esecuzione di un programma, il calcolatore può incontrare istruzioni che non riesce ad eseguire. Allora stampa un messaggio di errore e, a meno che questo non sia solo un avvertimento, il programma si ferma. Tutte le variabili del programma mantengono i valori che avevano al momento dell'errore. Il numero della linea che contiene l'errore viene stampato nel messaggio(ad esempio - CAN'T DO THAT IN 210).

*** BAD ARGUMENT**

1. Una funzione di TI BASIC ha un argomento non corretto
2. L'espressione di stringa delle funzioni ASC o VAL ha lunghezza zero (stringa nulla)
3. L'espressione di stringa nella funzione VAL non è una rappresentazione corretta di una costante numerica

*** BAD LINE NUMBER**

1. Il numero di linea specificato in una ON, GOTO o GOSUB non esiste nel programma
2. Il numero di linea specificato in una BREAK o in una UNBREAK non esiste nel programma (viene inviato solo un avvertimento)

*** BAD NAME**

1. Il nome del sottoprogramma in una CALL non è corretto

*** BAD SUBSCRIPT**

1. L'indice non è intero
2. L'indice ha un valore maggiore delle dimensioni specificate o consentite per una matrice
3. Usate un indice 0 quando nel programma c'è l'OPTION BASE 1

*** BAD VALUE**

CHAR

1. Nell'istruzione CHAR c'è un *codice carattere* fuori dall'intervallo consentito
2. Nell'istruzione CHAR c'è un carattere non valido nel *modello*

CHR$

3. La CHR$ ha un argomento minore di 0 o maggiore di 32767

COLOR

4. Il *numero dell'insieme dei caratteri* di un'istruzione COLOR è fuori dall'intervallo consentito
5. Il *codice di colore foreground* o *background* di un'istruzione COLOR è fuori dall'intervallo consentito

ELEVAMENTO A POTENZA(∧)

6. Avete elevato un numero negativo a un esponente frazionario

FOR

7. L'incremento di una FOR-TO-STEP è zero

HCHAR, VCHAR, GCHAR

8. In un'istruzione HCHAR, VCHAR, o GCHAR c'è un *numero di riga* o *di colonna* fuori dall'intervallo consentito

JOYST, KEY

9. L'*unità tastiera* di un'istruzione JOYST o KEY è fuori dall'intervallo consentito

ON

10. L'*espressione numerica* che individua un *numero di linea* è fuori dall'intervallo consentito

OPEN, CLOSE, INPUT, PRINT, RESTORE

11. Un *numero file* è minore di 0 o maggiore di 255
12. Il numero record nell'opzione SEQUENTIAL di un'istruzione OPEN è una variabile non numerica o maggiore di 32767
13. La *lunghezza record* dell'opzione FIXED di un'istruzione OPEN è maggiore di 32767

POS

14. L'*espressione numerica* in un'istruzione POS è negativa, zero o maggiore di 32767

SCREEN

15. Il *codice colore* del video è fuori dall'intervallo consentito

SEG$

16. Il valore dell'*espressione numerica 1* (posizione del carattere) o dell'*espressione numerica 2* (lunghezza della sotto-stringa) è minore di 0 o maggiore di 32767

SOUND

17. *Durata, frequenza, volume* o *disturbo* hanno valori fuori dagli intervalli consentiti

TAB

18. Il valore della posizione del carattere è maggiore di 32767 nella funzione TAB

*** CAN'T DO THAT**

1. C'è un RETURN senza una precedente GOSUB
2. C'è un NEXT senza il corrispondente FOR
3. La variabile di controllo nell'istruzione NEXT non è la stessa del FOR precedente
4. Il comando BREAK viene dato senza numero di linea

*** DATA ERROR**

1. Gli elementi di un'istruzione DATA non sono separati da virgole
2. La *lista variabili* di un'istruzione READ non è stata completamente esaurita e non ci sono altre istruzioni DATA nel programma
3. C'è un'istruzione READ senza una corrispondente DATA
4. In un'istruzione READ si assegna un valore di stringa a una variabile numerica
5. Il *numero di linea* in un'istruzione RESTORE è maggiore del più alto numero del programma

*** FILE ERROR**

1. Avete usato un'istruzione di CLOSE, INPUT, PRINT, o RESTORE per un file non aperto
2. Avete usato una INPUT per un file aperto in modo OUTPUT o APPEND
3. Avete usato una PRINT per un file aperto in modo INPUT
4. Avete usato una OPEN per un file già aperto

*** INCORRECT STATEMENT**

Casi generali

1. Manca una "(" o una ")" o entrambe
2. Manca una virgola
3. Manca il numero di linea in un'istruzione BREAK, UNBREAK o RESTORE (BREAK 100,)
4. "+" o "-" non sono seguiti da un'espressione numerica
5. Le espressioni usate con i segni aritmetici non sono numeriche

6. Le espressioni usate con gli operatori relazionali non sono dello stesso tipo7.
7. State tentando di usare una stringa come indice
8. State tentando di assegnare un valore a una funzione
9. C'è una parola riservata fuori posto10. E'
10. È presente un segno aritmetico o relazionale non previsto
11. Manca un segno aritmetico o relazionale

Sottoprogrammi di TI BASIC

12. Nella JOYST la *x di ritorno* e la *y di ritorno* non sono variabili numeriche
13. Nella KEY lo *stato della tastiera* non è una variabile numerica
14. In GCHAR la terza specifica deve essere una variabile numerica
15. Nella SOUND vi sono più di tre specifiche di tonalità o più di una specifica di disturbo
16. La CALL non è seguita da un nome di sottoprogramma

Istruzioni di Ingresso/Uscita per la gestione dei file

17. Mancano il simbolo "#" o i ":" nella specifica del *numero file* per la OPEN, CLOSE, INPUT, PRINT o RESTORE
18. Il *nome file* in una OPEN o DELETE deve essere un'espressione di stringa
19. In una OPEN manca una parola chiave o non è corretta
20. In una OPEN il numero dei record nella opzione SEQUENTIAL è minore di zero
21. La lunghezza record nell'opzione FIXED della OPEN è minore di zero o maggiore di 255
22. I ":" nella CLOSE non sono seguiti dalla parola chiave DELETE
23. Manca dove richiesto in una PRINT un separatore di stampa (,) (;) ...
24. Il *messaggio di richiesta* di una INPUT non è una espressione di stringa
25. Il *nome file* non è una stringa valida in un comando SAVE o OLD

Istruzioni generiche di programma

FOR

26. La parola chiave FOR non è seguita da una variabile numerica
27. Nella dichiarazione FOR la variabile di controllo non è seguita dal segno "="
28. Manca la parola chiave TO in una dichiarazione FOR
29. Nella dichiarazione FOR l'*estremo* non è seguito dalla fine della linea o dalla parola chiave STEP

IF

30. Manca la parola chiave THEN o non è seguita da un numero di linea

LET

31. Manca il segno "=" in una LET

NEXT

32. La parola chiave NEXT non è seguita da una variabile di controllo

ON-GOTO, ON-GOSUB

33. ON non è seguito da un'espressione numerica valida

RETURN

34. Un carattere o una parola non consentiti seguono la parola RETURN

Funzioni definite dall'utente

35. Il numero di argomenti di una funzione definita dall'utente non corrisponde al numero dei parametri

*** ERRORI DI INPUT**

1. Il dato di ingresso è troppo lungo per il buffer di I/O (se i dati sono introdotti da tastiera, si tratta solo di una segnalazione che permette di reintrodurre i dati)
2. Il numero di variabili nella *lista variabili* non corrisponde al numero di dati introdotti da tastiera o letti da un file (è una segnalazione solo per l'introduzione da tastiera)
3. Viene dato in ingresso un valore non numerico per una variabile numerica. Questa condizione può essere causata dalla lettura di caratteri di riempimento presenti in un record di un file.(E' una segnalazione solo per l'introduzione da tastiera)
4. Un dato numerico in ingresso dà overflow.(E' una segnalazione solo per l'introduzione da tastiera)

*** ERRORI DI INGRESSO/USCITA** - Questa condizione genera uno dei seguenti codici di errore

Messaggi di Errore

Quando si verifica un errore di I/O, viene visualizzato un codice di errore a due cifre insieme al messaggio
* I/O ERROR XY IN *numero di linea*
La prima cifra (X) indica quale operazione ha causato l'errore.

Valore di X	*Operazione*
0	OPEN
1	CLOSE
2	INPUT
3	PRINT
4	RESTORE
5	OLD
6	SAVE
7	DELETE

La seconda cifra (Y) indica il tipo di errore.
Valore di Y *Tipo di Errore*

0 Il nome del dispositivo in un comando DELETE, LIST, OLD o SAVE è errato
1 Tentate di scrivere su di un file protetto in scrittura
2 Una o più opzioni della OPEN sono errate o non corrispondono alle caratteristiche del file
3 Un comando di I/O non è valido
4 Tentate di scrivere su di un dispositivo di memoria con spazio insufficiente
5 Tentate di leggere oltre la fine del file
6 Un dispositivo non è collegato o è guasto. Questo errore può avvenire durante un'operazione su un file se il dispositivo esterno si è accidentalmente scollegato
7 Il file indicato non esiste, oppure il tipo del file (file di programma o file di dati) non corrisponde alla modalità di accesso

*** MEMORY FULL**

1. Non c'è abbastanza memoria per il carattere specificato nell'istruzione CHAR
2. L'istruzione GOSUB rimanda al suo stesso numero di linea
3. Il programma contiene troppi sottoprogrammi nidificati
4. Il programma contiene troppe funzioni definite dall'utente che ne richiamano altre, anch'esse definite dall'utente
5. Un'espressione numerica, relazionale o di stringa è troppo lunga
6. Una funzione definita dall'utente richiama se stessa

*** NUMBER TOO BIG** (viene data una segnalazione, e il valore viene sostituito con uno dei limiti previsti dal calcolatore)

1. Un'operazione numerica produce un overflow(valore $>$ di 9.9999999999999E127 o $<$ di -9.9999999999999E127)
2. Un dato letto con una READ da un'istruzione DATA causa un overflow
3. Un dato letto con una INPUT causa un overflow

*** STRING-NUMBER MISMATCH**

1. Viene specificato un argomento non numerico per una funzione di TI BASIC, una tab o un elevamento a potenza
2. Un valore non numerico si trova in una dichiarazione che lo richiede
3. Un valore non di stringa si trova in una dichiarazione che lo richiede
4. In una funzione definita dall'utente il tipo dell'argomento è diverso da quello del parametro, o il tipo della funzione è diverso da quello dell'espressione
5. Il *numero file* non è un valore numerico in una OPEN, CLOSE, INPUT O RESTORE
6. Tentate di assegnare una stringa a una variabile numerica
7. Tentate di assegnare un valore numerico ad una variabile di stringa

Nota - Si possono presentare altri codici di errore nell'uso di diversi dispositivi esterni, come il Sistema di Memoria a Dischi TI o la Stampante Termica. Consultate i relativi manuali per avere maggiori informazioni.

IV. Comando OLD

* CHECK PROGRAM IN MEMORY

Il comando OLD non cancella la memoria di programma se non quando l'operazione di caricamento va a buon fine. In caso contrario, qualsiasi programma in memoria può essere parzialmente o completamente ricoperto dal programma caricato. Listate il programma in memoria prima di proseguire.

Programmi applicativi

Introduzione

I programmi di questo paragrafo partono da livelli di estrema semplicità e diventano via via sempre più complessi. Potete iniziare da qualsiasi livello. La maggior parte di essi usano le prestazioni di grafica a colori e di produzione di suoni del vostro calcolatore, e questo vi fornisce una buona base per progettare poi da soli i vostri programmi grafici e sonori.

Esempi:

Bruco

Questo programma crea un bruco che si muove avanti e indietro sul video, emettendo un suono del tipo "uh-oh" quando urta un bordo, e tornando indietro nella direzione opposta.

```
>NEW
```

Potete dare un colore al bruco (sono consigliati i colori di codice 2-3,5-16). Il video viene cancellato. L'istruzione CALL COLOR assegna il colore scelto all'insieme di caratteri 2, e la variabile XDIR indica la direzione di movimento del bruco (+ 1 verso destra, — 1 verso sinistra).

```
>100 REM BRUCO
>110 CALL CLEAR
>120 INPUT "COLORE?":C
>130 CALL CLEAR
>140 CALL COLOR(2,C,C)
>150 XVECCHIO=1
>160 XDIR=1
```

Questo ciclo fa muovere il bruco sul video. La linea 180 calcola la posizione del blocco successivo, e la linea 190 lo visualizza sul video. Il ciclo T determina la velocità di movimento. La linea 220 cancella il blocco vecchio (per non tracciare una linea continua) sovrapponendo uno spazio sul blocco che si trova nella posizione XVECCHIO. La linea 230 contiene la posizione del blocco attuale, per poter calcolare la successiva. Il ciclo viene ripetuto finchè il bruco raggiunge il bordo del video.

```
>170 FOR I=1 TO 31
>180 XNUOVO=XVECCHIO+XDIR
>190 CALL HCHAR(12,XNUOVO,42)
>200 FOR T=1 TO 200
>210 NEXT T
>220 CALL HCHAR(12,XVECCHIO,32)
>230 XVECCHIO=XNUOVO
>240 NEXT I
```

La linea 250 inverte la direzione di movimento. Le linee 260 e 270 producono un suono "uh-oh", e la 280 rimanda in esecuzione il ciclo.

```
>250 XDIR=-XDIR
>260 CALL SOUND(100,392,2)
>270 CALL SOUND(100,330,2)
>280 GOTO 170
>RUN

--LO SCHERMO SI PULISCE

--LO SCHERMO SI PULISCE
```

Mosaico

Questo programma crea un mosaico sul video. I colori vengono prodotti in modo casuale, e ogni volta che si aggiunge una striscia di colore diverso viene prodotto un suono.

Esempi:

```
>NEW
```

Queste istruzioni cancellano lo schermo e assegnano un colore diverso ad ogni insieme di caratteri (da 2 a 16). L'istruzione RANDOMIZE serve per generare un diverso insieme di colori ogni volta che il programma viene eseguito.

```
>100 REM COMPOSIZIONE COLORI
>110 RANDOMIZE
>120 CALL CLEAR
>130 FOR S=2 TO 16
>140 CALL COLOR(S,S,S)
>150 NEXT S
```

Queste istruzioni producono un bordo per il mosaico

```
>160 CALL HCHAR(7,3,64,28)
>170 CALL HCHAR(16,3,64,28)
>180 CALL VCHAR(7,2,64,10)
>190 CALL VCHAR(7,31,64,10)
```

Questo ciclo aggiunge strisce di colore sul video da sinistra verso destra (dalla colonna 3 alla 30), emettendo ogni volta un suono. La durata negativa fa in modo che il suono venga interrotto e ne venga generato uno nuovo ogni volta che viene eseguita la CALL SOUND. Il sottoprogramma che inizia alla linea 310 genera colori e suoni casuali.

```
>200 FOR A=3 TO 30
>210 GOSUB 310
>220 CALL VCHAR(8,A,C,4)
>230 CALL SOUND(-150,Y,2)
>240 NEXT A
```

Questo ciclo è come quello delle linee 200-240, tranne che aggiunge le strisce colorate da destra verso sinistra, sotto quelle prodotte dal ciclo precedente. Alla fine del ciclo il programma torna alla linea 200 per ricominciare da sinistra.

```
>250 FOR A=30 TO 3 STEP -1
>260 GOSUB 310
>270 CALL VCHAR(12,A,C,4)
>280 CALL SOUND(-150,Y,2)
>290 NEXT A
>300 GOTO 200
```

Questo sottoprogramma genera un carattere casuale (e quindi un colore casuale) per le istruzioni CALL VCHAR(linee 220, 270). Le istruzioni di assegnamento delle linee 320 e 330 producono un suono di tonalità casuale. La RETURN rimanda il programma all'istruzione immediatamente successiva alla GOSUB (linee 210, 260).

```
>310 C=INT(120*RND)+40
>320 N=INT(24*RND)+1
>330 Y=220*(2^(1/12)^N
>340 RETURN
>RUN

--LO SCHERMO SI PULISCE

--APPARE LA COMPOSIZIONE

PREMETE CLEAR PER FERMARE IL PRO
GRAMMA
```

Il Gioco dei Numeri

Lo scopo di questo programma è di indovinare un numero scelto a caso tra 1 e un massimo introdotto da voi. Ad ogni tentativo voi introducete due numeri, e il calcolatore vi dice se il numero da indovinare è minore dei due, maggiore dei due o compreso fra i due. Quando pensate di avere indovinato introducete lo stesso numero due volte.

Esempi:

```
>NEW
```

L'istruzione RANDOMIZE assicura una diversa sequenza di numeri ogni volta che il programma viene eseguito. Le variabili MSG1$ e MSG2$ vengono ripetutamente usate nelle istruzioni PRINT, e la CALL CLEAR cancella il video.

```
>100 REM NUMERI SEGRETI
>110 RANDOMIZE
>120 MSG1$="IL NUMERO SEGRETO E'
 "
>130 MSG2$="I VOSTRI 2 NUMERI"

>140 CALL CLEAR
```

L'istruzione INPUT arresta il programma in attesa che voi introduciate il numero massimo. Poi viene generato il numero da indovinare e il video viene cancellato. La variabile N contiene il numero di tentativi fatti.

```
>150 INPUT "SCRIVETE IL LIMITE?
":L
>160 SEGR=INT(L*RND)+1
>170 CALL CLEAR
>180 N=N+1
```

Questa istruzione di INPUT vi chiede i due numeri. Se rispondete con due numeri uguali e indovinate il numero segreto, il programma passa alla linea 300. Se il numero è inferiore del minore dei due, passa alla linea 260. Se è maggiore del più alto, passa alla linea 280. Se è compreso tra i due o uguale a uno di loro, il programma continua.

```
>190 INPUT "ESTREMI INTERVALLO?
 ":BASSO, ALTO
>200 IF BASSO<>ALTO THEN 220
>210 IF SEGR=BASSO THEN 300
>220 IF SEGR<BASSO THEN 260
>230 IF SEGR>ALTO THEN 280
```

Queste istruzioni stampano un messaggio per comunicarvi in che relazione sono i vostri numeri con quello da indovinare. Poi il programma ritorna alla 180 e vi chiede di ritentare. Quando avete indovinato il programma vi comunica il numero dei vostri tentativi.

```
>240 PRINT MSG1$
>245 PRINT "COMPRESO TRA "&MSG2$
>250 GOTO 180
>260 PRINT MSG1$
>265 PRINT "MINORE DE"&MSG2$
>270 GOTO 180
>280 PRINT MSG1$
>285 PRINT "MAGGIORE DE"&MSG2$
>290 GOTO 180
>300 PRINT "AVETE INDOVINATO"
>310 PRINT "IN ";N;"TENTATIVI"
```

Il Gioco dei Numeri

Queste istruzioni vi offrono la possibilità di ripetere il gioco o di finire. Se introducete un carattere diverso da Y, il programma ha termine. Se volete giocare ancora, il contatore del numero dei tentativi viene azzerato, e il programma vi chiede se volete dare un altro valore massimo. Se rispondete S (SI) il programma torna alla linea 140. Con qualsiasi altro carattere, torna alla linea 160 per generare un nuovo numero da indovinare.

Esempi:

```
ESEMPI:

>320 PRINT "GIOCATE ANCORA?"
>330 INPUT "SCRIVETE S O N: ":A$
>340 IF A$<>"S" THEN 390
>350 N=0
>360 PRINT "DESIDERATE CAMBIARE
LIMITE?"
>370 INPUT "SCRIVETE S O N: ":B$
>380 IF B$="S" THEN 140 ELSE 160
>390 END
```

A destra vi mostriamo un'esecuzione del programma.(Naturalmente il numero che dovrete indovinare sarà diverso da quello dell'esempio.)

```
>RUN

--LO SCHERMO SI PULISCE

 SCRIVETE IL LIMITE? 20

--LO SCHERMO SI PULISCE

 ESTREMI INTERVALLO? 1,10
 IL NUMERO SEGRETO E'
 COMPRESO TRA I VOSTRI 2 NUMERI

 ESTREMI INTERVALLO? 1,5
 IL NUMERO SEGRETO E'
 MAGGIORE DEI VOSTRI 2 NUMERI



 ESTREMI INTERVALLO? 7,7
 AVETE INDOVINATO
 IN 3 TENTATIVI
 GIOCATE ANCORA?
 SCRIVETE S O N: N

 ** DONE **
```

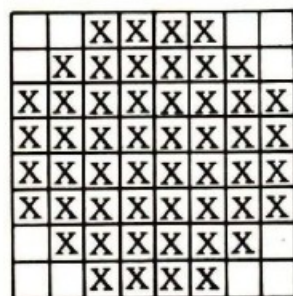
La pallina che rimbalza

Questo programma muove una pallina facendola rimbalzare sui bordi del video. Ogni volta che la pallina tocca un bordo rimbalza e emette un suono. Per definire la pallina si usano i seguenti caratteri speciali.

Esempi:

								Codici dei blocchi
		X	X	X	X			3C
	X	X	X	X	X	X		7E
X	X	X	X	X	X	X	X	FF
X	X	X	X	X	X	X	X	FF
X	X	X	X	X	X	X	X	FF
X	X	X	X	X	X	X	X	FF
	X	X	X	X	X	X		7E
		X	X	X	X			3C

```
>NEW
```

Queste istruzioni cancellano il video e definiscono la pallina col codice carattere 96.

```
>100 REM PALLINA CHE SALTA
>110 CALL CLEAR
>120 CALL CHAR(96,"3C7EFFFFFF
 FF7E3C")
```

Queste istruzioni vi chiedono il colore della pallina e quello di fondo del video. Notate che se definite il colore del video con l'insieme di caratteri 1, che comprende il carattere 32 (lo spazio) fissate dei limiti ben definiti. Il video viene cancellato quando avete introdotto i colori.

```
>130 INPUT "COLORE PALLINA? ":C
>140 INPUT "COLORE VIDEO? ":S

>150 CALL CLEAR
>160 CALL COLOR(9,C,S)
>170 CALL COLOR(1,S,S)
```

Queste istruzioni danno la posizione di partenza della pallina e definiscono i parametri che controllano le direzioni X e Y.

```
>180 X=16
>190 Y=12
>200 XDIR=1
>210 YDIR=1
```

Queste istruzioni calcolano la posizione successiva della pallina. La direzione del suo movimento dipende dai valori attuali di XDIR (+ 1 a destra, — 1 a sinistra) e YDIR (+ 1 in su, — 1 in giù).

```
>220 X=X+XDIR
>230 Y=Y+YDIR
```

Queste istruzioni controllano se sul video c'è tuttora la nuova posizione della pallina. Se il valore della riga (Y) o della colonna (X) sono fuori dall'intervallo consentito, il programma passa alla linea 310 (nel caso della colonna) o alla 360 (nel caso della riga) per cambiare la direzione della pallina.

```
>240 IF X<1 THEN 310
>250 IF X>32 THEN 310
>260 IF Y<1 THEN 360
>270 IF Y>24 THEN 360
```

La pallina che rimbalza

Se la nuova posizione della pallina è ancora sul video, questo viene cancellato per eliminare le vecchie posizioni, e la pallina viene poi visualizzata nella nuova posizione indicata da X e da Y.

Esempi:

```
>280 CALL CLEAR
>290 CALL HCHAR(Y,X,96)
>300 GOTO 220
```

Queste istruzioni cambiano la direzione della pallina quando X è fuori dall'intervallo consentito. L'istruzione CALL SOUND produce il suono del rimbalzo. Le linee 330 e 340 controllano se anche Y è fuori dall'intervallo. In questo caso il programma passa a cambiare anche la direzione di Y, altrimenti passa alla linea 220 per calcolare la nuova posizione.

```
>310 XDIR=-XDIR
>320 CALL SOUND(30,380,2)
>330 IF Y<1 THEN 360
>340 IF Y>24 THEN 360
>350 GOTO 220
```

Queste istruzioni cambiano la direzione della pallina se Y è fuori dall'intervallo. La CALL SOUND produce il suono del rimbalzo, e il programma passa alla linea 220 per calcolare la nuova posizione.

```
>360 YDIR=-YDIR
>370 CALL SOUND(30,380,2)
>380 GOTO 220
>RUN

--LO SCHERMO SI PULISCE

 COLORE PALLINA? 5
 COLORE VIDEO? 15

--UNA PALLINA COMPARE NEL CENTRO
 DELLO SCHERMO E COMINCIA A RIMB
ALZARE

PREMETE CLEAR PER FERMARE IL PRO
GRAMMA
```

Estratto Conto

In genere una volta al mese vi capita di confrontare la situazione dei vostri libretti di assegni con l'estratto conto della banca. Le due cose normalmente non coincidono, in quanto ci sono sempre prelievi o versamenti che non sono stati ancora registrati. Questo programma rende facile e veloce questa operazione.

Esempi:

```
>NEW
```

Queste istruzioni cancellano il video e vi chiedono di inserire il resoconto della vostra banca.

```
>100 REM ESTRATTO CONTO
>110 CALL CLEAR
>120 INPUT "ESTRATTO CONTO? ":B
```

Queste istruzioni vi dicono come introdurre i numeri e gli importi degli assegni che avete staccato. Potete usare indifferentemente PRINT o DISPLAY.

```
>130 DISPLAY "SCRIVETE I PRELIEV
I"
>140 DISPLAY "NUMERO E IMPORTO"
>150 DISPLAY
>160 DISPLAY "SCRIVETE ZERO PER
"
>170 DISPLAY "FINIRE I PRELIEVI"
>180 DISPLAY
```

Questo ciclo definisce la procedura per introdurre ogni numero e ogni importo. Questi valori sono memorizzati sotto forma di matrici. Se un numero è zero, il programma esce dal ciclo. CTOTAL contiene l'importo totale degli assegni. Ogni volta che introducete un importo, il programma salta alla linea 190 per ricevere un altro numero e il relativo importo.

```
>190 N=N+1
>200 INPUT "NUM. ASSEGNO? ":NA
>210 IF NA=0 THEN 250
>220 INPUT "IMPORTO? ":NI
>230 CTOT=CTOT+NI
>240 GOTO 190
```

Queste istruzioni vi dicono come introdurre i versamenti.

```
>250 DISPLAY "SCRIVETE I DEPOSIT
I"
>260 DISPLAY
>270 DISPLAY "SCRIVETE ZERO PER"
>280 DISPLAY "FINIRE I DEPOSITI"
>300 DISPLAY
```

Questo ciclo vi chiede i diversi versamenti. Se l'importo totale è zero, il programma esce dal ciclo. DTOTAL contiene l'importo totale dei versamenti. Dopo avere sommato un versamento al totale, il programma passa alla linea 310 per chiederne un altro.

```
>310 M=M+1
>320 INPUT "IMPORTO DEPOSITO? ":
DEP
>330 IF DEP=0 THEN 360
>340 DTOT=DTOT+DEP
>350 GOTO 310
```

Estratto Conto

Queste istruzioni calcolano e visualizzano il nuovo bilancio. Adesso potete introdurre i dati del vostro libretto di assegni, curando di sottrarre le spese per i servizi bancari. Infine viene calcolata e visualizzata la correzione necessaria per far quadrare il vostro libretto con l'estratto conto della banca.

Esempi:

```
>360 NB=B-CTOT+DTOT
>370 DISPLAY "NUOVO SALDO= ":NB
>380 INPUT "TOTALE CONTROLLO? ":
CONTR
>390 DISPLAY "CORREZIONE= ":NB-C
ONTR
>400 END
```

A destra c'è un esempio di esecuzione del programma.

```
>RUN

--LO SCHERMO SI PULISCE

 ESTRATTO CONTO? 940.26

 SCRIVETE I PRELIEVI
 NUMERO E IMPORTO

 SCRIVETE ZERO PER
 FINIRE I PRELIEVI

 NUM. ASSEGNO? 212
 IMPORTO? 76.83
 NUM. ASSEGNO? 213
 IMPORTO? 122.87
 NUM. ASSEGNO? 216
 IMPORTO? 219.50
 NUM. ASSEGNO? 218
 IMPORTO? 397.31
 NUM. ASSEGNO? 219
 IMPORTO? 231.00
 NUM. ASSEGNO? 220
 IMPORTO? 138.25
 NUM. ASSEGNO? 0
 SCRIVETE I DEPOSITI

 SCRIVETE ZERO PER
 FINIRE I DEPOSITI

 IMPORTO DEPOSITO? 450
 IMPORTO DEPOSITO? 0
 NUOVO SALDO= 204.5

 TOTALE CONTROLLO? 209.15
 CORREZIONE= -4.65

 ** DONE **
```

Giochi Grafici

Questo programma di giochi vi dà un esempio di come potete usare la grafica . Sono definiti sei diversi caratteri grafici - cuore, ciliegia, campana, limone, losanga e sbarra. Per giocare dovete semplicemente eseguire il programma. Il calcolatore genera tre numeri casuali tra 1 e 6, e sul video compaiono le tre figure corrispondenti. Il punteggio dipende da che cosa compare sul video. Dopo che sono state visualizzate le figure e il punteggio, potete giocare ancora.

Esempi:

Queste istruzioni definiscono il colore di ogni figura nel modo seguente.

```
>NEW

>100 REM GIOCHI GRAFICI
>110 CALL COLOR(9,7,16)
>120 CALL COLOR(10,13,16)
>130 CALL COLOR(11,2,16)
>140 CALL COLOR(12,6,16)
>150 CALL COLOR(13,11,16)
>160 CALL COLOR(14,5,16)
```

Caratteri grafici	*Colore*
Cuore	Rosso
Ciliegia	Rosso con gambo verde
Campana	Blu chiaro con anello nero
Limone	Giallo scuro
Losanga	Verde scuro
Sbarra	Blu scuro

Per tutte le figure lo sfondo è bianco.

```
>170 CALL CHAR(96,"00001C3E7F
 7F7F7F")
>180 CALL CHAR(97,"0000387CFE
 FEFEFE")
>190 CALL CHAR(98,"3F1F0F0703
 01")
>200 CALL CHAR(99,"FCF8F0E0C0
 80")
```

Codici Esadecimali																	*Codici Decimali*
00																	00
00																	00
1C				X	X	X					X	X	X				38
3E			X	X	X	X	X			X	X	X	X	X			7C
7F		X	X	X	X	X	X	X	X	X	X	X	X	X	X		FE
7F		X	X	X	X	X	X	X	X	X	X	X	X	X	X		FE
7F		X	X	X	X	X	X	X	X	X	X	X	X	X	X		FE
7F		X	X	X	X	X	X	X	X	X	X	X	X	X	X		FE
3F			X	X	X	X	X	X	X	X	X	X	X	X			FC
1F				X	X	X	X	X	X	X	X	X	X				F8
0F					X	X	X	X	X	X	X	X					F0
07						X	X	X	X	X	X						E0
03							X	X	X	X							C0
01								X	X								80
00																	00
00																	00

Notate che nelle linee 190 e 200 mancano gli ultimi quattro zeri. In questo modo si risparmia del tempo in quanto il calcolatore riempie automaticamente di zeri la parte di stringa restante.

Queste istruzioni descrivono la ciliegia.

Esempi:

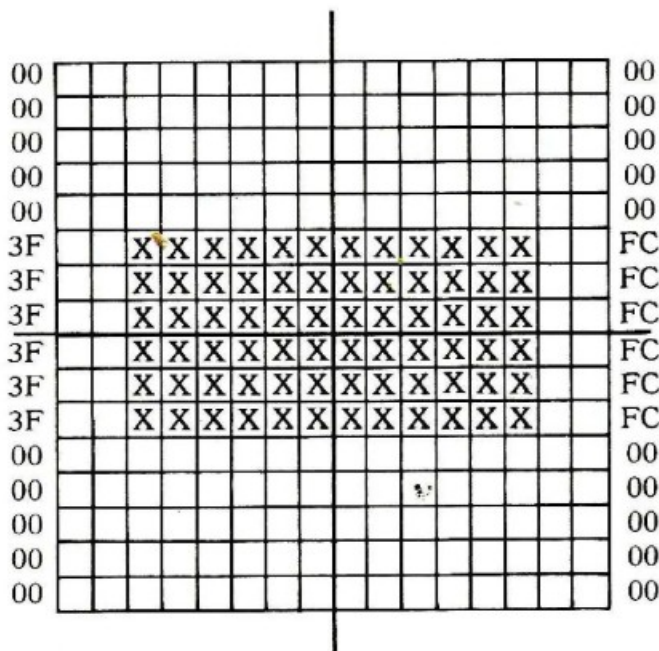
```
>210 CALL CHAR(100,"000000000
 01F3F7F")
>220 CALL CHAR(104,"000006081
 0204080")
>230 CALL CHAR(101,"7F7F7F7F3
 F3F1F")
>240 CALL CHAR(102,"E0F0F0F0F
 0E0C0")
```

Queste istruzioni descrivono la campana.

```
>250 CALL CHAR(112,"000001010
 1010101")
>260 CALL CHAR(113,"000080808
 0808080")
>270 CALL CHAR(120,"030707070
 70F0701")
>280 CALL CHAR(121,"C0E0E0E0E
 0F0E080")
```

Queste istruzioni descrivono la sbarra.

Esempi:

```
>370 CALL CHAR(136,"000000000
 03F3F3F")
>380 CALL CHAR(137,"000000000
 0FCFCFC")
>390 CALL CHAR(138,"3F3F3F")
>400 CALL CHAR(139,"FCFCFC")
```

L'istruzione RANDOMIZE serve per generare sequenze di figure sempre diverse ad ogni esecuzione del programma. La variabile C contiene la posizione della prima colonna della figura successiva. Il ciclo con variabile di controllo I genera un numero casuale tra 1 e 6 compresi. L'istruzione ON-GOSUB alla linea 460 trasferisce il programma al sottoprogramma che visualizza le figure sul video in base ai seguenti valori.

```
>410 RANDOMIZE
>420 CALL CLEAR
>430 C=14
>440 FOR I=1 TO 3
>450 PIC(I)=INT(6*RND)+1
>460 ON PIC(I) GOSUB 840,900,
 960,1020,1080,1140
>470 C=C+2
>480 NEXT I
```

PIC(I)	*Figura*
1	Cuore
2	Ciliegia
3	Campana
4	Limone
5	Losanga
6	Sbarra

Quando una figura è visualizzata sul video, il programma torna al ciclo che genera un nuovo numero e una nuova figura. Quando sul video ci sono tutte e tre le figure, si calcola il punteggio.

Queste istruzioni descrivono il limone.

Queste istruzioni descrivono la losanga.

Esempi:

```
>290 CALL CHAR(128,"000000030
 F1F3FFF")
>300 CALL CHAR(129,"000000C0F
 0F8FCFF")
>310 CALL CHAR(130,"FF3F1F0F0
 3")
>320 CALL CHAR(131,"FFFCF8F0C
 0")
```

```
>330 CALL CHAR(105,"000103070
 F1F3F7F")
>340 CALL CHAR(106,"0080C0E0F
 0F8FCFE")
>350 CALL CHAR(107,"7F3F1F0F0
 70301")
>360 CALL CHAR(108,"FEFCF8F0E
 0C080")
```

Queste istruzioni calcolano il vostro punteggio, in base alla tabella seguente. Il numero di linea è quello dell'istruzione in cui si fa il calcolo del punteggio relativo alla situazione descritta.

Partita	*Punti*	*Numero di linea*
Tre figure uguali	+ 75	700
Due ciliege, limoni o sbarre ai primi due posti	+ 40	550
Due cuori, losanghe o campane ai primi due posti	+ 10	650
Prima e ultima figura uguali	+ 10	650
Ultime due figure uguali	— 10	610

Esempi:

```
>490 REM PUNTEGGIO
>500 IF PIC(1)<>PIC(2) THEN 5
20
>510 IF PIC(2)=PIC(3) THEN 70
0 ELSE 540
>520 IF PIC(1)<>PIC(3) THEN 6
10
>530 GOTO 650
>540 IF PIC(1)/2<>INT(PIC(1)/
2) THEN 650
```

Queste istruzioni aggiungono 40 punti a quelli totalizzati. Viene prodotto un suono a tre tonalità e sul video compare un messaggio per segnalarvi che avete guadagnato 40 punti. Poi il programma passa alla linea 770 per visualizzare il punteggio totale.

```
>550 TOTAL=TOTAL+40
>560 CALL SOUND(100,440,2)
>570 CALL SOUND(100,660,2)
>580 CALL SOUND(100,550,2)
>590 PRINT "PREMIO--40 PUNTI"

>600 GOTO 770
```

Alla linea 610 vengono sottratti 10 punti dal punteggio totale. Un messaggio sul video ve lo comunica e si sente un suono a una tonalità. Il programma passa poi alla linea 770 per darvi il punteggio totale sul video.

```
>610 TOTAL=TOTAL-10
>620 CALL SOUND(100,110,2)
>630 PRINT "PERDI 10 PUNTI"
>640 GOTO 770
```

Queste istruzioni aggiungono 10 punti al punteggio, producono un suono bitonale e un messaggio opportuno. Il programma passa poi alla linea 770 per darvi il punteggio totale.

```
>650 TOTAL=TOTAL+10
>660 CALL SOUND(100,660,2)
>670 CALL SOUND(100,770,2)
>680 PRINT "GUADAGNI 10 PUNTI"
>690 GOTO 770
```

Queste istruzioni aggiuntono 75 punti al totale, producono un suono a cinque tonalità e un messaggio per dirvi che avete vinto.

```
>700 TOTAL=TOTAL+75
>710 CALL SOUND(100,440,2)
>720 CALL SOUND(100,550,2)
>730 CALL SOUND(100,440,2)
>740 CALL SOUND(100,660,2)
>750 CALL SOUND(100,880,2)
>760 PRINT "EVVIVA--75 PUNTI"
```

L'istruzione PRINT alla linea 770 stampa il vostro attuale punteggio. Le altre vi offrono la possibilità di fare un'altra partita o di smettere. L'istruzione CALL KEY alla linea 800 accetta la vostra risposta senza che dobbiate premere **ENTER**. Con una Y dite al programma di tornare alla linea 410 per generare altre tre figure. Con qualsiasi altro tasto fate finire il programma.

```
>770 PRINT "PUNTEGGIO ATTUALE: "
;TOTAL
>780 PRINT "VUOI GIOCARE ANCORA?
"
>790 PRINT "SCRIVI S PER CONTINU
ARE"
>800 CALL KEY(0,KEY,STATUS)
>810 IF STATUS=0 THEN 800
>820 IF KEY=83 THEN 410
>830 END
```

Giochi grafici

Questi sei sottoprogrammi stampano una delle sei figure. La RETURN serve a stampare una solo figura ogni volta che il sottoprogramma viene chiamato.

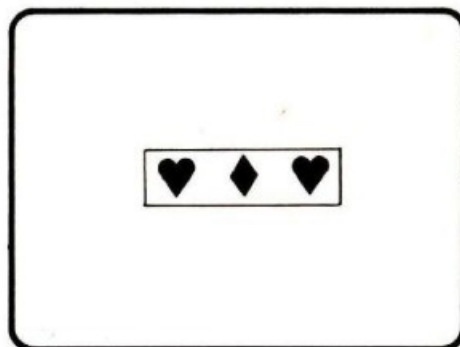
Esempi:

```
>840 REM DISEGNO CUORE
>850 CALL HCHAR(12,C,96)
>860 CALL HCHAR(12,C+1,97)
>870 CALL HCHAR(13,C,98)
>880 CALL HCHAR(13,C+1,99)
>890 RETURN
>900 REM DISEGNO CILIEGIA
>910 CALL HCHAR(12,C,100)
>920 CALL HCHAR(12,C+1,104)
>930 CALL HCHAR(13,C,101)
>940 CALL HCHAR(13,C+1,102)
>950 RETURN
>960 REM DISEGNO PALLA
>970 CALL HCHAR(12,C,112)
>980 CALL HCHAR(12,C+1,113)
>990 CALL HCHAR(13,C,120)
>1000 CALL HCHAR(13,C+1,121)
>1010 RETURN
>1020 REM DISEGNO LIMONE
>1030 CALL HCHAR(12,C,128)
>1040 CALL HCHAR(12,C+1,129)
>1050 CALL HCHAR(13,C,130)
>1060 CALL HCHAR(13,C+1,131)
>1070 RETURN
>1080 REM DISEGNO LOSANGA
>1090 CALL HCHAR(12,C,105)
>1100 CALL HCHAR(12,C+1,106)
>1110 CALL HCHAR(13,C,107)
>1120 CALL HCHAR(13,C+1,108)
>1130 RETURN
>1140 REM DISEGNO SBARRA
>1150 CALL HCHAR(12,C,136)
>1160 CALL HCHAR(12,C+1,137)
>1170 CALL HCHAR(13,C,138)
>1180 CALL HCHAR(13,C+1,139)
>1190 RETURN
```

A destra vi diamo un esempio di esecuzione del programma. Notate che il video resta viola mentre il calcolatore crea la tavola dei simboli e scorre il programma per trovare gli errori. Il tutto circa in un minuto.

Esempi:

```
>RUN

--LO SCHERMO SI PULISCE
```

```
--SUONO DI DUE NOTE
```

```
GUADAGNI 10 PUNTI
PUNTEGGIO ATTUALE: 10
VUOI GIOCARE ANCORA?
SCRIVI S PER CONTINUARE
```

```
SCRIVI S PER CONTINUARE N

 ** DONE **
```

Terminologia e definizioni

ASCII - American Standard Code for Information Interchange. E' il codice usato per rappresentare internamente lettere, numeri e caratteri speciali dalla maggior parte dei "personal computer".

Base 100 - sistema di numerazione con base 100. Vedi il paragrafo "Costanti Numeriche" per la rappresentazione dei numeri.

BASIC - semplice linguaggio di programmazione usato dalla maggior parte dei "personal computer". BASIC è un acronimo che significa "Beginners All-purpose Symbolic Instruction Code".

Baud - unità di misura della velocità di trasmissione. 1 baud=1 bit al secondo.

Binario - sistema di numerazione che usa due cifre, 0 e 1. Su di esso si basano il linguaggio macchina e le operazioni dei calcolatori.

Buffer - area di memoria destinata alla memorizzazione temporanea di un record in ingresso o in uscita.

Bug - malfunzionamento hardware o errore di programmazione che provoca un'esecuzione non corretta di un'operazione.

Byte - una stringa di cifre *binarie* (bit) che in genere rappresenta un *carattere*. Spesso viene usato come unità di misura della capacità di memoria di un calcolatore. Ad esempio, un calcolatore con 16K byte di memoria dispone di circa 16.000 byte di memoria per programmi e dati.

Carattere - una lettera, un numero, un simbolo di punteggiatura o un simbolo grafico speciale.

Carattere di controllo - simbolo(>)che segna l'inizio di un *comando* o di una *linea di programma*. Simbolo o frase che richiede dati di ingresso all'utente.

Ciclo - un gruppo di linee di programma che viene eseguito più volte.

Comando - istruzione che il calcolatore esegue immediatamente. I comandi non fanno parte di un programma, e vengono quindi introdotti senza numero di linea.

Concatenazione - unione di due o più *stringhe* per formarne una più lunga, tramite l'operatore di concatenazione &.

Costante - valore numerico o di *stringa*. Una costante numerica è un numero reale, come 1.2 o -9054. Una costante di stringa è una combinazione di al massimo 112 caratteri tra virgolette, come ECCOCI QUA" o "VIA ROMA 275".

Cursore - simbolo che indica la posizione in cui comparirà sul video il prossimo *carattere* introdotto da tastiera.

Dato - elemento base di informazione elaborato o prodotto dal calcolatore.

Default - caratteristica o valore standard che il calcolatore assume in mancanza di dichiarazione diversa in un'istruzione o in un programma.

Disco - supporto per memoria di massa ad accesso diretto e sequenziale.

Dispositivo - (vedi *Dispositivo esterno*)

Dispositivo esterno - apparecchiatura accessoria che si collega al calcolatore ed estende le sue prestazioni. Appartengono a questa categoria i *Moduli di comando* pre-programmati e le unità di trasmissione, ricezione e memorizzazione dei dati, come stampanti e dischi, che vengono anche chiamate periferiche.

Disturbo - suoni diversi che vengono usati per produrre particolari effetti sonori. Il *sottoprogramma* CALL SOUND produce un *disturbo* invece che una tonalità quando viene specificato un valore negativo per la frequenza(da -1 a -8).

Esadecimale - sistema di numerazione in base 16 che usa 16 simboli, da 0 a 9 e da A a F. Viene usato per una rappresentazione più immediata del codice *binario*. Ad esempio, 1010 in binario è uguale ad A in esadecimale, 11111111 è uguale a FF. Il codice Esadecimale è usato nel sottoprogramma CALL CHAR per la costruzione delle sagome dei caratteri grafici.

Eseguire - eseguire un programma. Eseguire l'operazione indicata da un'*istruzione* o da un *comando*.

Esponente - numero che indica la potenza a cui viene elevato un numero o un'*espressione*. Di solito viene scritto in alto a destra del numero. Per esempio, 2^8=2*2*2*2*2*2*2*2. In TI BASIC l'esponente è preceduto dal simbolo o dalla lettera E nella *notazione scientifica*. Per esempio, 2^8=2∧8; 1.3*x10^{25}=1.3E25.

Terminologia e definizioni

Espressione - combinazione di costanti, variabili e operatori che porta a un unico risultato. Può essere numerica, di stringa o relazionale.

File - insieme di record di dati memorizzati su di un dispositivo. Si usa anche per indicare quei *dispositivi* di ingresso/uscita non strutturati a file, come la stampante.

Fine del File (End-of-file) - segnalazione che tutti i record di un file sono stati letti.

Formato interno dei dati - rappresentazione dei *dati* all'interno del calcolatore. I dati numerici occupano 8 *byte* più uno che contiene la lunghezza. Le stringhe occupano un byte per carattere più uno per la lunghezza.

Funzione - tramite le funzioni si definiscono come operazioni "singole" procedure diverse, ognuna delle quali comporta un certo numero di passi, ad esempio l'estrazione di radice.

Grafica - costruzione di grafici, sagome e disegni sul video, sia fissi che in movimento. TI BASIC possiede sottoprogrammi interni di sistema che consentono facili applicazioni di grafica a colori.

Hardware - i diversi dispositivi che costituiscono il sistema del calcolatore, comprese le memorie, la tastiera, il video, le unità a dischi, le stampanti, etc.

Hertz (Hz) - unità di misura della frequenza . 1 Hertz=1 ciclo al secondo.

Incremento - valore positivo o negativo che modifica sensibilmente una *variabile*.

Input - (sostantivo = ingresso) *dato* da introdurre nella memoria del calcolatore - (verbo=introdurre) procedura di caricamento di un dato in memoria.

Intero - numero intero, positivo, negativo o nullo.

I/O - Ingresso/Uscita(Input/Output). In genere indica la funzione di un certo dispositivo, e viene usato per la comunicazione tra calcolatore e dispositivi esterni (tastiera, dischi, etc.)

Iterazione - tecnica di ripetizione di un gruppo di istruzioni di un programma. Una ripetizione del gruppo. Vedi *ciclo*.

Linea - vedi *linea grafica, linea di ingresso, linea di stampa, linea di programma.*

Linea di ingresso - quantità di *dati* che possono essere introdotti di seguito. In TI BASIC sono 112 caratteri.

Linea di programma - una linea contenente una *istruzione*. Può avere al massimo 112 *caratteri*.

Linea di stampa - linea di 28 caratteri usata dalle istruzioni PRINT e DISPLAY.

Linea grafica - linea di 32 caratteri usata dai sottoprogrammi di grafica del TI BASIC.

Mantissa - base di un numero espresso in *notazione scientifica*. Nel numero 3.264 + 4 la *mantissa* è 3.264.

Matrice - insieme di variabili numeriche o di stringa organizzate in una lista o in una tabella per essere elaborate dal calcolatore. La posizione di un elemento all'interno di una matrice è individuata da un *indice*.

Memoria - vedi *RAM, ROM, memoria di massa.*

Memoria di massa - *dispositivo esterno* come un registratore a cassette o un'unità a dischi, che conserva programmi e/o *dati* che il calcolatore deve usare in seguito. Le informazioni sono in genere in un formato leggibile dal calcolatore ma non dall'uomo.

Modo Comando - quando non sta eseguendo alcun programma, il calcolatore è in Modo Comando (o Immediato), ed esegue qualsiasi comando immediatamente.

Edit - è il modo usato per modificare linee di programma. Si entra in Modo Edit con il comando Edit. La linea specificata compare sul video e se ne può modificare qualsiasi *carattere* con i tasti per l'"editing".

Modo Immediato - vedi *Modo Comandi.*

Modo Number - è il modo in cui si trova il calcolatore quando genera automaticamente numeri di linee da introdurre o da modificare.

Modulo - vedi *Moduli di Comando.*

Moduli di Comando - moduli di programma, registrati in ROM, che si possono facilmente introdurre nel calcolatore per estendere le sue prestazioni.

Terminologia e definizioni

Numero pseudo-casuale - numero prodotto con una sequenza definita di calcoli (algoritmo), abbastanza casuale da poterlo considerare tale per certe elaborazioni. Un autentico numero casuale si ottiene solo per caso.

Operatore - simbolo usato nei calcoli (operatore numerico) o in espressioni relazionali (operatore relazionale). Gli operatori numerici sono +, —, *, /, ∧. Gli operatori relazionali sono >, <, =, >=, <=, <>.

Output - (sostantivo = uscita) informazione fornita dal calcolatore- (verbo = produrre un'uscita) trasferire informazioni dalla memoria del calcolatore su un dispositivo esterno, come il video, la stampante o una *memoria di massa.*

Overflow - condizione che si verifica quando viene dato in ingresso o prodotto da un calcolo un valore maggiore di 9.9999999999999 E127 o minore di - 9.9999999999999 E127. Questo valore viene sostituito da uno dei valori estremi previsti dal calcolatore, viene inviato un avviso e il *programma* prosegue l'esecuzione.

Parametro - valore che deteremina o modifica il risultato di un'*istruzione* o di una *funzione.*

Programma - insieme di istruzioni che guidano il calcolatore nell'esecuzione di un compito particolare.

Punto di arresto (Breakpoint) - è un punto di un programma, specificato dal comando BREAK, in cui l'esecuzione deve essere sospesa. Durante un breakpoint potete eseguire operazioni in *Modo Comandi* per individuare eventuali errori nel programma. Si può far ripartire il programma con il comando CONTINUE, se non è stato modificato.

RAM - Random Access Memory (memoria ad accesso diretto). è la memoria principale del calcolatore, dove vengono conservati temporaneamente dati e *istruzioni* durante l'*esecuzione* di un programma. Programmi e dati su RAM possono essere letti, utilizzati e modificati, e vengono persi quando si spegne il calcolatore o si esce dal BASIC.

Record - insieme di dati correlati tra loro, come ad esempio le informazioni sullo stipendio di un dipendente, o i voti degli esami di uno studente. Un insieme di record contenenti lo stesso tipo di informazioni, come gli stipendi di tutti i dipendenti di un'azienda, si chiama *file.*

Record a lunghezza fissa - record di un file che hanno tutti le stessa lunghezza. Se un file ha record di lunghezza fissa pari a 95 caratteri, ogni record occupa 95 *byte*, anche se contiene *dati* solo per 76 posizioni. Il calcolatore aggiunge caratteri di riempimento alla fine del record per fargli raggiungere la lunghezza specificata.

Salto - procedura che modifica l'esecuzione sequenziale delle istruzioni di un programma. Un salto incondizionato trasferisce il controllo alla linea di programma specificata ogni volta che viene eseguito. Un salto condizionato lo trasferisce in base al risultato di alcune operazioni aritmetiche o logiche.

Stringa nulla - una *stringa* che non contiene alcun carattere e ha lunghezza zero.

Video - schermo.

Visualizzare - far comparire caratteri sul video.